Network Layer

- Transport Layer (Layer 4)
  - Application-to-application communication
- Network Layer (Layer 3)
  - Host-to-host communication
- Route
  - The path that a packet takes through the network
- Routing
  - The process of moving the packet along the route
- Forwarding
  - Transferring a packet from an incoming link to an outgoing link
- Router
  - The device that forwards packets (datagrams)

Forwarding vs. Routing

- Routing
  - Responsibility over the path
  - Routing algorithms figure out the path a packet should take
- Forwarding
  - A router consults a forwarding table
  - Examines data in a packet's header & uses the table to determine the outgoing link for the packet
  - Routing algorithms configure forwarding tables
- Switches vs. Routers
  - Packet switches: transfer data between links based on link layer data (e.g., Ethernet)
  - Routers: transfer data between links based on network layer data (e.g., IP)

Network service models: our wish list

- Guaranteed delivery (no loss)
- Bounded (maximum) delay
- In-order packet delivery
- Guaranteed constant or minimum bandwidth
- Maximum jitter
  - Jitter = variation in latency
- Endpoint authentication & encrypted delivery

Network service models: what do we get?

- IP gives us none of this
  - Best-effort = no guarantees on delivery, delay, order
- Other network architectures provide some of these items
  - E.g., ATM (Asynchronous Transfer Mode)
  - ATM CBR (Constant Bit Rate)
    - Connection setup specifies bandwidth
    - Network provides constraints on jitter and packet loss
    - Network guarantees in-order delivery
  - ATM ABR (Available Bit Rate)
    - In-order delivery
    - Guaranteed minimum bandwidth but higher rates if resources available
    - Feedback to sender if congestion is present

Virtual Circuit vs. Datagram Networks

- Virtual Circuit (VC) Networks
  - Connection service at the network layer
  - All routers in the path are involved in the connection
- Datagram Networks
  - Connectionless service at the network layer
  - Connection-oriented service provided at the transport layer
  - Only end systems are involved
  - Routers are oblivious

*IP is a datagram network*
Virtual Circuit Networks

- Connection setup
  - Set up route based on destination address
  - Each router commits resources
  - Each router builds entries in its forwarding table
    - Routers maintain connection state information

- Communication
  - Each packet contains a VC#
    - Forwarding table determines the next link and VC#
    - Destination address not needed on each packet; just the VC#

- Teardown
  - Clear connection from forwarding table on each router

Datagram Networks

- Packet identified with the destination address
- No setup; routers maintain no state information
- Routers
  - Use the destination address to forward the packet
  - Forwarding table maps destination address to output link
- IP addresses are 32 bits
  - We can't have a forwarding table with $2^{32}$ (4,294,967,296) entries!
  - Match a range of addresses by matching a prefix

Incoming Interface

<table>
<thead>
<tr>
<th>Incoming Interface</th>
<th>Source</th>
<th>Outgoing Interface</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>101</td>
</tr>
</tbody>
</table>

Routing Protocols

Prefix    | Destination Interface
----------|-----------------------
192.168.**.* = Rutgers
0000-0001 0000-0110 2
0000-0111 0011 3
0000-0111 0110 1

Incoming Interface

<table>
<thead>
<tr>
<th>Incoming Interface</th>
<th>Source</th>
<th>Outgoing Interface</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>101</td>
</tr>
</tbody>
</table>

Router Architecture

- Control Plane: Routing & Management
  - Run routing protocols & maintain routing tables
  - User command interface
  - Accounting
  - ICMP
  - Queue management

Data Plane: Packet Forwarding

- Layer 1: Retrans & regenerate signal
- Layer 2: Rewrite header and checksum
- Layer 3: Look up, queue, decrement TTL, regenerate checksum, forward to output port

Note: a port on a router refers to the input & output interfaces, not a transport layer port

Router Architecture: line cards

A line card is responsible for I/O on a specific interface

Shared Memory - Conventional

- Ports
  - Function as I/O devices in an OS
- Packet arrival
  - CPU interrupt
  - Copied to memory
- Routing
  - CPU determines route
  - Copies packet to output port
- Limitation
  - Only one memory read/write at a time
  - CPU & bus can be bottlenecks
**Shared Memory – Distributed CPUs**

- CPU & copy of routing table in each line card
- Lookup and data copy to output port done by line card
- Limitation
  - Only one memory read/write at a time
  - Bus can be a bottleneck

**Shared Bus – No Shared Memory**

- No shared memory
- Bus used to copy packets directly from one port to another
- Limitation
  - Shared bus can be a bottleneck

**Non-shared Memory – Crossbar Data Path**

- N×N crossbar switching fabric
- One port can move a packet to another port without blocking other ports
- Multiple switching fabrics can be used to route packets to the same port
- Verdict
  - Fastest solution
  - $$$

**Output Port Queuing**

- If there’s a queue at an output port
  - A packet scheduler chooses one packet for transmission
  - This can be simple first-come-first-served (FCFS)
  - ... or take other factors into account (source, destination, protocol, service level)
- If the output port queue is full
  - We have packet loss
  - A router can decide which packet to drop
  - Active Queue Management (AQM) algorithms: decide which packets to drop

**Input Port Queuing**

- If packets arrive faster than they could be switched
  - They need to be queued at input ports
  - If multiple queues have a packet for the same output port
    - Only one will be switched at a time
    - The others will be blocked ... and the packets behind them will be blocked too!
    - Head-of-line blocking
  - If the queue overflows
    - We have packet loss

**Head-of-line blocking**

- If this packet has to wait
  - Then these packets have to wait
Internet Protocol

**IP Datagram Structure**
- 20 byte fixed part
- Variable-size options

**IP Datagram: Version**
- 4-bit identification of the protocol used: 4 = IPv4

**IP Datagram: Header Length**
- 4-bit header length (in # of 32-bit words)
  - IP packets usually have no options, so this is usually 5

**IP Datagram: DSCP**
- Differentiated Services Control Point
  - Identifies class of service for QoS aware routers (e.g., VoIP)
Explicit Congestion Notifications
- Routers normally do not inform endpoints of congestion
- ECN is an optional feature to allow them to do so

Identification
- If TTL == 0, discard the packet
- Keeps packets from circulating indefinitely (common TTL = 60...64)

Fragmentation
- Identification: Identifies fragment of an original datagram
- Flags: control fragmentation or identify if there are more fragments
- Fragment offset: offset of fragment relative to original data

Header Checksum
- 1's complement checksum of the header
- Router discards packet if corrupted
- Must be recalculated by the router since TTL (& maybe options) change
**IP Datagram: Source & Destination**

- Identifies source and destination IP addresses

**IP Datagram: Options**

- Extensions to the header – rarely used
- Options include: route to destination, record of route, IP timestamp

**IP Fragmentation & Reassembly**

- When an IP datagram is first created
  - Sender creates an ID number for each datagram (usually value of a counter)
  - DF bit (“Don’t Fragment”) set to 0: fragmenting is allowed

- When a router needs to fragment a datagram
  - Each fragment contains the same ID #, source address, destination address
  - Fragment offset
  - Identifies offset of the fragment relative to the original datagram in 8-byte blocks
  - All fragments except for the last one have the MF (“More Fragments”) bit set

**Example: send 4,000 byte datagram**
- 20 bytes IP header + 3980 bytes data
- Outbound link at router has a 1500-byte MTU

<table>
<thead>
<tr>
<th>Source IP address</th>
<th>Destination IP address</th>
<th>Options (if header length = 1)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.36.211.59</td>
<td>128.6.4.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID = 2222</td>
<td>TTL = 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum = xxx</td>
<td>Offset = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF = 0</td>
<td>Data = 3980 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len = 4000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source IP address</th>
<th>Destination IP address</th>
<th>Options (if header length = 1)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.36.211.59</td>
<td>128.6.4.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID = 2222</td>
<td>TTL = 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum = ahh</td>
<td>Offset = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF = 1</td>
<td>Data = 1480 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len = 1500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source IP address</th>
<th>Destination IP address</th>
<th>Options (if header length = 1)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.36.211.59</td>
<td>128.6.4.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID = 2222</td>
<td>TTL = 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum = ccc</td>
<td>Offset = 185</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF = 0</td>
<td>Data = 1480 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len = 1500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source IP address</th>
<th>Destination IP address</th>
<th>Options (if header length = 1)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.36.211.59</td>
<td>128.6.4.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID = 2222</td>
<td>TTL = 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum = ccc</td>
<td>Offset = 370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF = 0</td>
<td>Data = 1020 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len = 1040</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IP Reassembly**

- Identification
  - Receiver knows a packet is a fragment if MF is 1 and/or Fragment Offset is not 0

- Matching & Sequencing
  - Identification field is used to match fragments from the same datagram
  - Offsets identify the sequence of fragments

- Size of original
  - When the receiver gets the last fragment (MF==0, Offset != 0)
  - It knows the size of the datagram (offset + length)

- Giving up
  - If any parts are missing within a time limit, discard the packet
  - Linux: `/proc/sys/net/ipv4/intradm tse` (default 30 seconds)
  - Once reassembled, pass to protocol that services this datagram

**Remember MTU (Maximum Transmission Unit)?**
- Maximum size of payload that a link-layer frame can carry
- This limits the size of an IP datagram (and hence a TCP or UDP segment)

- What if a router needs to forward a packet that is larger than that link’s MTU?
  - Break up the datagram into two or more fragments
  - Each fragment is a separate IP datagram
  - IP layer at the end system needs to reassemble the fragments before passing the data to the transport layer
### IP Addressing

- **IPv4 address**: 32 bits expressed in dotted-decimal notation
  - Example: `www.rutgers.edu = 01001010011000000100000011101001 = 128.6.68.137

- Each interface needs to have an IP address
  - E.g., each link on a router has an address
  - If your laptop is connected via Ethernet and 802.11, you have 2 IP addresses
  - Every interface at a router has its own address

### Route Aggregation: Subnets

- **IP address**: 32 bits = $2^{32}$ addresses
  - But addresses cannot be assigned randomly
  - Otherwise routing tables would have to be $2^{32}$ entries long!
  - ... and maintaining them would be a nightmare

- Instead, assign groups of adjacent addresses to an organization

- **Subnet** (network)
  - A logical network connected to a router (LAN or collection of LANs)

- **Rutgers subnet**: 128.6.0.0/16
  - CIDR notation (Classless Inter-Domain Routing)
  - A/N: N most significant (leftmost) bits of address

- **Rutgers subnet**: 128.6.0.0/16
  - First 16 bits of the IP address identify a host at Rutgers
  - Routers need to know how to route to just 128.6 instead of all 65,536 ($2^{16}$) possible addresses

- **Route aggregation**: use one prefix to advertise routes to multiple devices or networks

### Subnet Mask

- **A subnet mask** (or netmask)
  - A bit mask with 1s in the network number position
    - Address & netmask → strips away host bits
  - Address & ~netmask → strips away network bits

- **For Rutgers**, the netmask is 255.255.0.0

### How are IP addresses assigned?

- **IP addresses are distributed hierarchically**
  - Internet Assigned Numbers Authority (IANA) at the top
  - IANA is currently run by ICANN

- **Regional Internet Registries (RIR)**
  - Allocate blocks of addresses to ISPs

- **Allocate blocks of addresses to end users**

- **Your computer** (or Internet gateway)
  - We will look at NAT later

- **Internet Corporation for Assigned Names and Numbers**
  - Internet Corporation for Assigned Names and Numbers

- **NAT**
  - Internet Assigned Numbers Authority (IANA) at the top
  - IANA is currently run by ICANN

- **Regional Internet Registries (RIR)**
  - Allocate blocks of addresses to ISPs

- **Allocate blocks of addresses to end users**

- **Your computer** (or Internet gateway)
  - We will look at NAT later

- **Internet Corporation for Assigned Names and Numbers**
  - Internet Corporation for Assigned Names and Numbers

- **NAT**
Address allocation: it's a hierarchy

- Allocates blocks of IP addresses
- Allocates one or more blocks of IP addresses
- Allocates an address or one or more blocks of IP addresses

Routing: dividing a network into smaller networks, can be repeated at each level of the hierarchy

Subnet Mask Example Within Rutgers

- Rutgers = 128.6.0.0 – netmask is 255.255.0.0
  - IP address range: 128.6.0.0 – 128.6.255.255

- Rutgers iLab systems are on a subnet within Rutgers
  - IP address range: 128.6.13.128 – 128.6.13.255

Special addresses

- Network address: all host bits 0
  - Rarely, if ever, used
  - Rutgers = 128.6.0.0

- Limited broadcast address: all bits 1
  - Broadcast address for this network, the local network.
  - Datagrams are not forwarded by routers to other networks

- Directed broadcast address: all host bits 1
  - All hosts on the specified subnet get datagrams sent to this address
  - Routers may or may not forward broadcasts (no for outside an organization)
  - Rutgers iLab systems = 128.6.13.255 (network=128.6.13.128)

- Loopback address: 127.0.0.1 = localhost
  - Communicate with your own device
  - Uses the loopback network interface

Host Configuration

- How do you assign an address to a host?
  - Manually, configure the device with its
    - IP address
    - Subnet mask, so it knows what addresses are local
    - Gateway: default address for non-local addresses not in a routing table
    - Routers that connects the LAN to another network
  - Automatically, via the Dynamic Host Configuration Protocol (DHCP)

Dynamic Host Configuration Protocol

- Protocol for client to get an IP address and network parameters

  - It has to work before the client has a valid address on the network!
  - Use IP broadcasts
  - DHCP server must be running on the same network link (LAN)
  - Else each link must run a DHCP Relay Agent that forwards the request to a DHCP server

DHCP: Three mechanisms for allocation

1. Automatic allocation
   - DHCP assigns an permanent IP address to a client
   - This association remains fixed until the administrator changes it

2. Dynamic allocation
   - DHCP assigns an IP address to a client for a limited period of time
   - Allows automatic reuse of an address that is no longer needed by the client

3. Manual allocation
   - A client IP address is assigned by the network administrator
DHCP: The Protocol

Client broadcasts DHCP Discover
  • Client sends a limited broadcast DHCP Discover UDP message to port 67
  • Contains random transaction identifier

Server responds with an offer
  • Server sends a limited broadcast DHCP Offer UDP message to port 68
  • Response contains
    – Matching transaction identifier
    – Proposed IP address
    – Subnet mask
    – Lease time

Client broadcasts DHCP Request
  • Sends back a DHCP message with a copy of the parameters
  • This performs selection (if multiple offers), confirmation of data, extension of lease

Server sends DHCP ACK
  • Sends configuration parameters, including committed IP address

NAT: Network Address Translation

• Every device on the Internet needs an IP address
  – Every address has to be unique
  … otherwise, how do you address a host?

• IP addresses are not plentiful
  – Does an organization with 10,000 IP hosts really need 10,000 addresses?

NAT: Private Addresses

• We cannot use IP addresses of valid external hosts locally
  … how will we distinguish local vs. external hosts?

• RFC 1918: Address Allocation for Private Internets
  – Defines unregistered, non-routable addresses for internal networks

<table>
<thead>
<tr>
<th>Address Range</th>
<th># addresses</th>
<th>CIDR block</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0 – 10.255.255.255</td>
<td>16,777,216</td>
<td>10.0.0.0/8</td>
</tr>
<tr>
<td>172.16.0.0 – 172.31.255.255</td>
<td>1,048,576</td>
<td>172.16.0.0/12</td>
</tr>
<tr>
<td>192.168.0.0 – 192.168.255.255</td>
<td>65,536</td>
<td>192.168.0.0/16</td>
</tr>
</tbody>
</table>

NAT variants

• Static NAT
  – One-to-one mapping between internal and external addresses

• Dynamic NAT
  – Maps an unregistered (internal) IP address to one of several registered IP addresses

• Overloading, or Port Address Translation (PAT)
  – A form of dynamic NAT that maps multiple internal IP addresses to a single registered address by using different ports

NAT: Network Address Translation

• Private IP address space in the organization
  – One external IP address

• NAT Translation Table
  – Map source address/port in outgoing IP requests to a unique external address/port
  – Inverse mapping for incoming requests

  A NAT-enabled router looks like a single device with one IP address

Translation Table in a NAT-Enabled Router

<table>
<thead>
<tr>
<th>Internal</th>
<th>Port</th>
<th>External</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.32.4</td>
<td>1200</td>
<td>68.36.210.55</td>
<td>4000</td>
</tr>
<tr>
<td>192.168.32.6</td>
<td>1200</td>
<td>68.36.210.55</td>
<td>4001</td>
</tr>
<tr>
<td>192.168.32.8</td>
<td>1100</td>
<td>68.36.210.55</td>
<td>4003</td>
</tr>
<tr>
<td>192.168.32.11</td>
<td>9000</td>
<td>68.36.210.55</td>
<td>4000</td>
</tr>
</tbody>
</table>

Translation Table in a NAT-Enabled Router

• NAT requires a router to look at the transport layer!
  – Source port (outgoing) & destination port (incoming) changes
  – TCP/UDP checksum recomputed

NAT variants

• Static NAT
  – One-to-one mapping between internal and external addresses

• Dynamic NAT
  – Maps an unregistered (internal) IP address to one of several registered IP addresses

• Overloading, or Port Address Translation (PAT)
  – A form of dynamic NAT that maps multiple internal IP addresses to a single registered address by using different ports
## Advantages of NAT

- Internal address space can be much larger than the addresses allocated by the ISP
- No need to change internal addresses if ISP changes your address
- Enhanced security
  - A computer on an external network cannot contact an internal computer
  - Unless the internal computer initiated the communication
  - But can only contact the computer on that specific port (this is where active mode FTP had problems)

## Internet Control Message Protocol (ICMP)

- Network-layer protocol to allow hosts & routers to communicate network-related information
- ICMP information is carried as IP payload

### ICMP Segment Structure

- Variable-size segment; 8-byte minimum
- Type: command or status report ID
- Code: status code for the type
- Checksum: Checksum from ICMP header & data
- Rest of header: depends on type
  - Error reports contain the IP header & first 8 bytes of original datagram's data

### Some ICMP Message Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>Destination unreachable</td>
</tr>
<tr>
<td>4</td>
<td>Source quench</td>
</tr>
<tr>
<td>5</td>
<td>Redirect message</td>
</tr>
<tr>
<td>8</td>
<td>Echo request</td>
</tr>
<tr>
<td>9</td>
<td>Router advertisement</td>
</tr>
<tr>
<td>10</td>
<td>Router solicitation</td>
</tr>
<tr>
<td>11</td>
<td>TTL exceeded</td>
</tr>
<tr>
<td>12</td>
<td>Bad IP header</td>
</tr>
<tr>
<td>13</td>
<td>Timestamp</td>
</tr>
<tr>
<td>14</td>
<td>Timestamp reply</td>
</tr>
<tr>
<td>15</td>
<td>Address mask request</td>
</tr>
<tr>
<td>18</td>
<td>Address mask reply</td>
</tr>
</tbody>
</table>

### Ping program

- Get a network ping (echo) from a requested host
  - Test network reachability
  - Measure round-trip time
  - Optionally specify packet size

- Request/response protocol
  - Ping Client
    - Create socket (AF_INET, SOCK_RAW, IPPROTO_ICMP)
    - Set IP header fields & ICMP header fields
    - Send it to a destination via sendto()
    - Wait for a response from the destination address via recvfrom()
Ping program

- **Request**
  - Send ICMP type=8 (echo request), code 0 (no options to echo)

- **Reply**
  - Destination responds back with an ICMP type=0 (echo reply), code 0

Traceroute program

- **Traceroute** – trace a route to a specific host
  - Send a series of UDP segments with a bogus destination port
  - 33434 to 33534 on Linux systems
  - First IP datagram has TTL=1
  - Second IP datagram has TTL=2, and so on
  - Keep a timer for each datagram sent

  - At a router
    - When the TTL expires, a router sends an ICMP warning message
      - Type 11, code 0 = TTL expired
      - ICMP message includes the name of the router and its IP address
  - At the final destination
    - The destination sends an ICMP warning message
      - Type 3 code 3 = Destination port unreachable

IPv6

- We’ve been rapidly using up IPv4 addresses ever more rapidly
  - Growth of the web
  - Always-on IP devices
  - Set-top boxes and phones
  - Inefficient network allocation
- We dealt with it with
  - NAT
  - Name-based web hosting
  - Reallocation of network allocation & subnetting
- Those solutions helped a lot... but not enough
  - We’re out of IPv4 addresses in parts of the world
    - ARIN’s free pool of IPv4 address space was depleted on September 24, 2015
    - IPv6 to the rescue!

### Highlights

- **Huge address space**
  - 128-bit addresses: $3.4 	imes 10^{38}$ addresses (>7.9 x 10^{28} more than IPv4)
- **Simplified 40-byte header**
  - Longer addresses but far fewer fields
  - Focus is to simplify routing
- **Anycast address**
  - Allows a datagram to be delivered to one of a group of interfaces
  - Usually used to identify the nearest host of several hosts
- **Flow label**
  - Allows related packets that require specific levels of service to be identified
  - E.g., voice, video
  - Not well defined yet

### IP Datagram Structure

- **Version**: protocol version = 6
- **Traffic class**: defines a category of service
- **Flow label**: identification tag for related flows
- **Payload length**: # bytes following the 40-byte datagram
- **Next header**: identifies higher-level protocol (e.g., TCP or UDP)
  - Same as in IPv4
  - Also permits extensions to IPv6, such as fragmentation, authentication
- **Hop limit**: TTL; decremented at each router
- **Source & destination addresses**
  - Data
- **No fragmentation** – need to use IPv6 extension headers
  - Routers will never fragment IPv6 datagrams!
- **No header checksum! Ethernet does it; so do TCP and UDP**
Transitioning

- IPv6 systems can bridge to IPv4 networks
  - IPv4 addresses are a subset of IPv6 addresses
- Dual-stack systems
  - Hosts with both IPv4 and IPv6 network stacks to communicate with both protocols
  - DNS can identify if a given domain is IPv6 capable or not
- IPv4 systems cannot communicate with IPv6 systems
  - Migrating to IPv6 results in a loss of global visibility in the IPv4 network
- Initial transition is not visible to end users
  - Cable modems, set-top boxes, VoIP MTAs
  - IPv6 access