Outline

- IP Packet Format
- IPv6
- NAT

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
IPv4 Header Fields

- Identifier, flags, fragment offset → used primarily for fragmentation
- Time to live
  - Must be decremented at each router
  - Packets with TTL=0 are thrown away
  - Ensure packets exit the network
- Protocol
  - Demultiplexing to higher layer protocols
  - TCP = 6, ICMP = 1, UDP = 17...
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – 16 bit
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues
    - Poorly supported

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... jokes... ...are always... told in parts...

- 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, fragment packets
Reassembly

- Where to do reassembly?
  - End nodes or at routers?

- End nodes
  - Avoids unnecessary work where large packets are fragmented multiple times
  - If any fragment missing, delete entire packet

- Dangerous to do at intermediate nodes
  - How much buffer space required at routers?
  - What if routes in network change?
    - Multiple paths through network
    - All fragments only required to go through destination

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

- MTU = 4000

- Length = 3820, M=0

- IP Header
- IP Data

IP Fragmentation Example #2

- MTU = 2000

- Length = 3820, M=0

- IP Header
- IP Data

- 3800 bytes

- IP Header
- IP Data

- 1980 bytes

- IP Header
- IP Data

- Length = 1840, M=0, Offset = 1980

- 1820 bytes
IP Fragmentation Example #3

Fragmentation and Reassembly Concepts

- Demonstrates many Internet concepts
- Decentralized
  - Every network can choose MTU
- Connectionless
  - Each (fragment of) packet contains full routing information
  - Fragments can proceed independently and along different routes
- Best effort
  - Fail by dropping packet
  - Destination can give up on reassembly
  - No need to signal sender that failure occurred
- Complex endpoints and simple routers
  - Reassembly at endpoints

IP Reassembly

- Fragments might arrive out-of-order
  - Don’t know how much memory required until receive final fragment
- Some fragments may be duplicated
  - Keep only one copy
- Some fragments may never arrive
  - After a while, give up entire process

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
    - Suggest alternate routing path for future messages
  - Redirect
    - Helps newly connected host discover local router
  - Time-to-live reached
    - Suggest alternate routing path for future messages

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 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

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

Outline

- IP Packet Format
- IPv6
- NAT

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)

IPv6 Cleanup - Router-friendly

- Common case: Switched in silicon ("fast path")
- Weird cases: Handed to CPU ("slow path", or "process switched")
  - Typical division:
    - Fast path: Almost everything
    - Slow path:
      - 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.
      - If it became popular, they’d be in the soup!
- Other speed issue: Touching data is expensive.
  - Designers would like to minimize accesses to packet during forwarding.

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
  - 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

- IP Packet Format
  - IPv6
  - NAT

Altering the Addressing Model

- Original IP Model
  - Every host has a unique IP address
- Implications
  - Any host can find any other host
  - 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
  - Packet traffic observable by routers and by LAN-connected hosts
  - Possible to forge packets
    - Use invalid source address
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

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
  - Firewall
    - Doesn’t let any packets from internal node escape
    - Outside world doesn’t need to know about internal addresses

Reducing IP Addresses

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

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

<table>
<thead>
<tr>
<th>Int Addr</th>
<th>Int Port</th>
<th>NAT Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2.2.2</td>
<td>1000</td>
<td>5000</td>
</tr>
</tbody>
</table>

NAT: Server Response

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

<table>
<thead>
<tr>
<th>Int Addr</th>
<th>Int Port</th>
<th>NAT Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2.2.2</td>
<td>1000</td>
<td>5000</td>
</tr>
</tbody>
</table>

NAT: Enabling Servers

- Use port mapping to make servers available

<table>
<thead>
<tr>
<th>Int Addr</th>
<th>Int Port</th>
<th>NAT Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3.3.3</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>
- Manually configure NAT table to include entry for well-known port
- External users give address 243.4.4.80
- Requests forwarded to server

Properties of Firewalls with NAT

- Advantages
  - Hides IP addresses used in internal network
  - Easy to change ISP: only NAT box needs to have IP address
  - Fewer registered IP addresses required
  - Basic protection against remote attack
  - Does not expose internal structure to outside world
  - Can control what packets come in and out of system
  - Can reliably determine whether packet from inside or outside

- Disadvantages
  - Contrary to the “open addressing” scheme envisioned for IP addressing
  - Hard to support peer-to-peer applications
  - Why do so many machines want to serve port 1214?
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 at 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.

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

Next Lecture

- How do forwarding tables get built?
- Routing protocols
  - Distance vector routing
  - Link state routing