The Internet Protocol (IP) – Part 1 (CSE 573S)

Ken Wong
Washington University
kenw@wustl.edu
www.arl.wustl.edu/~kenw

Internets And The Internet

- An Internet: A network of heterogeneous networks
- THE (Global) Internet
  - An internet that uses IP
  - Organized into a multilevel hierarchy
- An Internet-Capable Host
  - Has a 32-bit IP address
    - e.g., 128.192.64.10, 0x80C0400A
  - Formats data into IP packets
  - Knows how to route packets to their destination
- Goals of Internetworking
  - Universal connectivity
  - Uniform access (hide hardware/software heterogeneity)

Basic Internet Technology

- Packets
  - Packets carry information and are self-describing
  - A packet has 2 parts:
    - A header (metadata (information about the payload))
    - A payload (information content)
- Store-And-Forward Technology
  - The metadata allows a packet to be stored at a router for eventual delivery
    - The packet can be released when convenient
  - Direct analogy with the post office system
  - Less expensive to operate than the telephone network

Internet Protocol Layers

- Services at 1 level depend on lower layer services
- The layers form a protocol stack
**Internet Protocol Layers**

- **Physical**
  - Interface between data transmission device and medium

- **Network**
  - Accessing and routing across the same network
  - Exchange data between end system and network
  - End system addressing

- **Internet (IP)**
  - Routing between different networks
  - End system addressing that hides network heterogeneity

- **Transport (UDP, TCP)**
  - Process addressing (Port number)
  - Reliable, ordered delivery

- **Application**

**IP Service Model**

- Datagram (Connectionless) data delivery model
  - Best-effort: No guarantee of datagram delivery
    - Unreliable, unordered delivery; Duplicate datagram service
    - Simplifies job of routers
    - End systems provide reliable, ordered delivery
  - Connectionless: No connection setup phase
    - Datagram has self-describing header

- **Addressing scheme**
- Leads to "hour glass" architecture with IP at the narrowest point
  - IP can run over any technology

**Internetworking Overview**

**Internet Architecture**

- **FTP**
- **HTTP**
- **NV**
- **TFTP**
- **Ethernet**
- **FDDI**
**IPv4 Packet**

- **Version (4):** Number of 32-bit words in header
  - Minimum is 5
- **Header Length (4):** Number of 32-bit words in header
  - Minimum is 5
- **Type of Service (8):** 3-bit Precedence, 4-bit ToS, 1-bit ignored
- **Total Length (16):** # bytes in IP datagram (includes header)
  - Maximum is 65535 but physical network may support much less
- **Identification (8):** Incremented for each datagram
- **Flags (3):** Don’t fragment; M(ore)
- **Fragment Offset (13):** 8-byte offset of fragment
- **Time to Live (8):** Became upper bound on number of hops
- **Protocol (8):** Demultiplexing field
- **Header Checksum (8):** Internet checksum over header

**Transmitted in big-endian byte order (NBO)**

---

**Path MTU Discovery**

- **600-Byte UDP, DF=1**
  - ICMP can’t fragment (return IP header)

- **Send pkt with DF=1**
- **If pkt > MTU, ICMP "can’t fragment" error pkt returns (optional: MTU that caused problem)**
  - If MTU is not returned, sender has to guess a new MTU

---

**Fragmentation Example**

- **Length = 512 bytes**
  - Offset = 512
  - M-bit = 1
  - All frags have same identifier

- **Length = 376 bytes**
  - Offset = 1024
  - M-bit = 0
  - All frags have same identifier
Reassembly

- Reassembly is done at the receiver
- All fragments except the last one have M-bit set
- Last fragment has M-bit cleared
- All fragments have the same header identifier
- The (i+1)th fragment has offset = sum all lengths of preceding fragments
- Need a fragment list data structure for holding fragments until all have arrived to receiver
- A periodic process garbage collects fragments after a timeout
- If a fragment is lost, the whole pkt is dropped
- Use "path MTU discovery" to avoid fragmentation

IPv4 ADDRESSES

- Router
  » A host with an interface on more than one network
  » Default Route: “Near-by” versus distant network
- Some Special IP Addresses
  » Field of all Os means "this" (Restrictions apply)
    - Network 0 in source network number means this network
    - Host 0 in source network number means this host
  » Directed Broadcast: Host Id = all 1s
  » Limited Broadcast (never forwarded): 32 1-bits
  » Network 127 is loopback network number (loop back to sender)

IPv4 Addressing

- A unique address for each active interface
  » A central authority allocates blocks of IP addresses to organization

IP Address Example

- Gateway (Border Router)
  - 128.252.254.1 From the Internet

- R10
- R8
- R6
- R4
- R2
- R3
- H

- Network 127 is loopback network number (loop back to sender)
- Hierarchic Addresses:
  - Reduces routing table size (aggregation)
IP Addressing Problems

- Apparently rigid hierarchic IP addressing scheme
  - Very similar IP addresses are on same physical network
    - 128.252.153.* (e.g., 128.252.153.16, 128.252.153.33)
  - Almost similar IP addresses are near each other (hop-wise)
    - 128.252.*.* (e.g., 128.252.169.6, 128.252.153.2)
  - Class A, B, C networks
- Unused address blocks in Class A network
- Large number of very small networks
  - Administrative cost of managing address space
  - Large routing tables (50,000 entries not uncommon)
  - IP address space exhaustion

Weaknesses Of IP Addressing

- Move host far enough ➔
  - Change IP address
- Run out of IP addresses ➔
  - Change Netid to larger class net
  - Reconfigure all hosts on the network
- Multihomed Hosts ➔
  - Different host name (IP Address) may mean different behavior
- Three Changes Since 1984 (Bandages)
  - Subnetting
  - CIDR (Classless InterDomain Routing)
  - DHCP (Dynamic Host Configuration Protocol)

LAN Routing Example

- IP pkt (with header) doesn’t change during transit
  - IP Src = 156.33.1.130, IP Dst = 156.33.1.1
- Ethernet header is modified
  - Src and Dst addresses reflect hop-by-hop transit
    - (e3.s → e2.e), (e2.w → e1.s)

IP Routing (Basic Idea)

- Given an IP pkt, get pkt 1 hop closer to destination
  - Router doesn’t need to know the entire path
- IP Lookup Function
  - Dst IP Address → Address of next hop interface
- Ethernet LAN
  - Determine hardware addresses (8:0:20:8e:19:5e) of src and dst interfaces
- Subnetting
  - Partition network address space into subnet address spaces
  - Result is Hierarchic Addressing and Hierarchic Routing
  - Accomodates growth: Router doesn’t need to know much about distant destinations
  - Details of how to split up local part of address left to network manager
  - Con: Difficult to change hierarchy once the structure is chosen
Routing Wish List

- Fast IP lookup ⇒ Small or "well-structured" routing tables
  - Border gateway has 2 entries (7 entries???)
  - R3 has 3 entries (???)
- Efficiently route packets
  - Minimize number of hops
- Simple router management
  - Don’t need to know the path to all hosts, just subnets
  - Standard IP address class hierarchy produces large routing table at gateway!!
  - Departments manage details of their own network plants
    - 192.168.0.0 is a private network accessible only from 128.252.10.0
  - Avoid need for massive IP address space reorganization
    - Gracefully handles growth in address space usage

Subnet Addressing

- Required part of IP addressing
  - RFC 950, RFC 1122
- Subnet Addressing

<table>
<thead>
<tr>
<th>Internet Part</th>
<th>Physical Net</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>129.2.0.0</td>
<td></td>
</tr>
<tr>
<td>Subnet 3.0</td>
<td>129.2.3.5</td>
<td>129.2.3.21</td>
</tr>
<tr>
<td>Subnet 7.0</td>
<td>Subnet 129.2.7.0</td>
<td></td>
</tr>
</tbody>
</table>

Subnet Masks

- A subnet mask indicates with 1's the network part and with 0's the host part
  - Example (Hex): 0xfffffff0 0
- Representations
  - Hexadecimal: 0xfffffff0 0
  - Dotted Decimal: 255.255.0.0
  - 3-tuples: {-1,-1,0} (network, subnet, host)

<table>
<thead>
<tr>
<th>11111111 11111111 11111111 00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 8 . 2 5 2 . 1 5 3 . 1 0</td>
</tr>
</tbody>
</table>

Network  Subnet  Host

Subnet Mask Usage

- Extracting Net/Subnet Id and Host Id
  ```
  uint32_t netmask, ipDst, network, host; // or in_addr_t
  network = netmask & ipDst;
  host = (~ netmask) & ipDst;
  ```

- Match Destination IP Address With Route Entry
  ```
  if ((routeEntry->netmask & dgram->dst) == routeEntry->dst) {
    ... Route entry matches destination IP address ...
  }
  ```
Routing Algorithm (1)

Route (Dgram dg, RouteTbl rt) { // Datagram, Routing Table
  D = Extract destination IP address from dg;
  if (D matches any directly connected network address) {
    Physical Address = Resolve(D); // ARP
    I = Determine outgoing interface;
    Encapsulate and Send dgram over interface I;
  } else {
    foreach (<Network,Mask,Router> entry in rt)
      if (D matches entry in rt)
        Encapsulate and Send dgram to Router;
  }
  // Should have matched default route in route table
  if (no matches) Routing Error;
}

Routing Algorithm (2)

- Match: Compare bitwise AND of dst IP address and netmask with network address
- Idea: Allow arbitrary netmasks ➔ Handle special cases in general way
  - Special Cases: Default route, host-specific route
- Route to a specific host
  - Netmask 255.255.255.255, Network address = Host IP address
- Default Route
  - Netmask 0.0.0.0, Network address = 0.0.0.0
- Standard Class B network without subnets
  - Netmask 255.255.0.0

Routing Example (1)

<table>
<thead>
<tr>
<th>e.w</th>
<th>Internet</th>
<th>R0</th>
<th>156.33.0.130</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e1.s</th>
<th>156.33.0.131</th>
</tr>
</thead>
<tbody>
<tr>
<td>e2.w</td>
<td>156.33.1.129</td>
</tr>
<tr>
<td>R1</td>
<td>156.33.1.2</td>
</tr>
<tr>
<td>e2.e</td>
<td>R2</td>
</tr>
<tr>
<td>e3.s</td>
<td>R3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e1.s</th>
<th>156.33.1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>e2.e</td>
<td>R2</td>
</tr>
<tr>
<td>e3.s</td>
<td>R3</td>
</tr>
</tbody>
</table>

- Subnet 1 or 156.33.0.128
- Subnet 1 or 156.33.1.128
- Subnet 3 or 156.33.1.130

Netmask = Ox ff ff ff 80 = 255.255.255.128

Routing Example (1)

- Address Ranges of Class A, B, C Networks
  - A: 00... = 0.0.0.0
  - B: 10... = 2^7 ➔ 128.0.0.0
  - C: 110... = ... + 2^6 = ... + 64 ➔ 192.0.0.0
  - D: 1110... = ... + 2^5 = ... + 32 ➔ 224.0.0.0
  - E: 11110... = ... + 2^4 = ... + 16 ➔ 240.0.0.0

- Number of Networks and Hosts
  - A: 128, 2^4
  - B: 2^16 - 128, 2^16
  - C: 2^24 - 128 - 2^16, 2^8
Routing Example (2)

- Network 156.33.0.0 = 0x 9C 11 00 00
  - 9C = 1001 110 = Class B network
  - Class B = 16-bit Network and 16-bit Host
- Netmask = 0xffffffff80 = 255.255.255.128 = 9-bit subnet; 7-bit host
  - 128 subnets, 128 hosts per subnet (Approximately)

Address Ranges
- Subnet 1: 156.33.0.128 - 156.33.0.255 (0..0 10..0 - 0..0 11..1)
- Subnet 2: 156.33.1.0 - 156.33.1.127 (0..1 10..0 - 0..1 11.1)
- Subnet 3: 156.33.1.128 - 156.33.1.255 (0..1 00..0 - 0..1 11..1)

Consider host 156.33.0.139
- Netmask AND (IP address) = 255.255.255.128 AND 156.33.0.139 = 156.33.0.128
- Netmask AND (IP address) = 0.0.0.127 AND 156.33.0.139 = 0.0.0.11

Routing Tables

<table>
<thead>
<tr>
<th>Entry</th>
<th>ID</th>
<th>Mask</th>
<th>Next Hop</th>
<th>Interface</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0[0]</td>
<td>156.33.0.128</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>e.s</td>
<td>Subnet 1</td>
</tr>
<tr>
<td>R0[1]</td>
<td>156.33.1.0</td>
<td>255.255.255.128</td>
<td>156.33.0.131</td>
<td>w.s</td>
<td>Subnet 2</td>
</tr>
<tr>
<td>R0[2]</td>
<td>156.33.1.128</td>
<td>255.255.255.128</td>
<td>156.33.0.131</td>
<td>w.s</td>
<td>Subnet 3</td>
</tr>
<tr>
<td>R0[3]</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>Internet</td>
<td>e.w</td>
<td>Internet</td>
</tr>
<tr>
<td>R1[0]</td>
<td>156.33.0.128</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>e1.n</td>
<td>Subnet 1</td>
</tr>
<tr>
<td>R1[1]</td>
<td>156.33.1.0</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>e1.s</td>
<td>Subnet 2</td>
</tr>
<tr>
<td>R1[2]</td>
<td>156.33.1.128</td>
<td>255.255.255.128</td>
<td>156.33.1.2</td>
<td>w1.s</td>
<td>Subnet 3</td>
</tr>
<tr>
<td>R1[3]</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>156.33.0.130</td>
<td>w1.n</td>
<td>Default</td>
</tr>
<tr>
<td>R2[0]</td>
<td>156.33.1.0</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>w2.w</td>
<td>Subnet 2</td>
</tr>
<tr>
<td>R2[1]</td>
<td>156.33.1.128</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>w2.s</td>
<td>Subnet 3</td>
</tr>
<tr>
<td>R2[2]</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>156.33.1.1</td>
<td>w2.w</td>
<td>Default</td>
</tr>
<tr>
<td>R3[0]</td>
<td>156.33.1.128</td>
<td>255.255.255.128</td>
<td>DIRECT</td>
<td>w3.s</td>
<td>Subnet 3</td>
</tr>
<tr>
<td>R3[1]</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>156.33.1.129</td>
<td>w3.s</td>
<td>Default</td>
</tr>
</tbody>
</table>

Example 1

- R3 (Src = 156.33.1.130) sends IP packet to R1 (Dst = 156.33.1.1)
  - At R3
    - Dst does not match interface IP address
    - R3[0]: 156.33.1.1 & 255.255.255.128 → 156.33.1.0 (No Match)
    - R3[1]: 156.33.1.1 & 0.0.0.0 → 0.0.0.0 (MATCH!!!)
      * Route to 156.33.129 (Out interface e3.s)
  - At R2
    - Dst does not match interface IP address
    - R2[0]: 156.33.1.1 & 255.255.255.128 → 156.33.1.0 (MATCH!!!)
      * Route directly (Out interface e2.w)
  - At R1
    - Dst matches interface IP address 156.33.1.1 → Deliver to IP

Address Resolution Protocol (ARP)

Broadcast:

Unicast:
ARP Implementation

- Request for binding (IA → PA):
  - Search ARP cache
  - Broadcast ARP request and wait for reply
    - Broadcast has PA and IA of sender and IA of destination
    - Reply can be delayed (busy host) or never received (down host)
    - Buffer outgoing packet that triggered ARP request
    - Release buffer when reply is returned or a timeout occurs
    - Handle ALL outstanding ARP requests for the same destination
    - Stale ARP cache value (age cached values; i.e., soft state)
  - Update ARP cache
  - Process packets waiting for IA → PA binding

- Entire subnet reads IA → PA request
  - Cache broadcaster’s IA → PA mapping
  - Send ARP reply message to broadcaster if receiver is the ARP target

ARP Implementation Issues

- Target host may be down or too busy to accept request
- Request can be lost because Ethernet provides a best-effort service
- Stale ARP cache entry
  - e.g., host ethernet interface is replaced
  - Cache entry has soft state; i.e., entry is removed if timer expires

Optimizations

- Address Resolution Cache (Cache IA → PA mappings)
- Piggyback broadcaster’s IA-PA binding onto the broadcast message
- All hosts on the broadcast network can cache the broadcaster’s IA → PA binding

ARP Message Format

- Encapsulated in Frame

<table>
<thead>
<tr>
<th>Hardware Type</th>
<th>Protocol Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hlen</td>
<td>Plen</td>
</tr>
<tr>
<td>Sender HA</td>
<td>Sender IP</td>
</tr>
<tr>
<td>Sender IP</td>
<td>Target HA</td>
</tr>
<tr>
<td>Target HA</td>
<td>Target IA</td>
</tr>
</tbody>
</table>

ARP Protocol Format

- No fixed format for ARP messages; depends on network technology
- Header indicates field lengths
- Ethernet ARP/RARP Message Format
  - HARDWARE TYPE (1 → Ethernet)
  - PROTOCOL TYPE (x0800 → High-level addresses are in IP format)
  - HLEN: Hardware address length
  - PLEN: Protocol address length
  - OPERATION: (1) Request or (2) Reply
  - SENDER HA, IP: Sender’s hardware and IP addresses
  - TARGET HA, IP: Target’s hardware and IP addresses
- ARP requestor supplies SENDER HA, IP, and TARGET IP
- Replier fills in TARGET HA: swaps SENDER and TARGET
ARP Example

- R3 needs PA for 156.33.1.129, the next hop interface
  - R3 broadcasts ARP request to find out PA(156.33.1.129)
  - R2.e2.e sends ARP reply to R3.e3.s (unicast, not a broadcast)
  - Now R3 knows the binding of 156.33.1.130 to e2.e!

- Alternative: Gratuitous ARP
  - During boot process, every host sends an ARP request for its own IP address
    - Effectively announces its own IA -> PA binding

Internet Control Message Protocol

- Allows IP systems to send error and administrative messages
- Required part of any IP implementation
- Usage
  - Errors: Routers report problems (e.g., can't route datagram; congestion)
  - Queries: Defined in request/reply pairs
    - e.g., Hosts test reachability (ping)
  - ICMP is an error reporting (not correction) mechanism
    - Error message is sent to the datagram source
    - Can not be used to directly inform intermediate routers of a problem; e.g.
      - Router Rk in path "R1, R2, ..., Rj, Rk" detects a routing problem
      - Rj has a bad routing table ... Rk can only tell R1 there was an error

ICMP Message Delivery

- An ICMP message is encapsulated in an IP datagram
- Datagram protocol field = 1 ➔ Message is carried in an IP datagram
- Applications send/receive ICMP messages through raw IP interface
- ICMP messages that cause an error are silently dropped

ICMP Redirect Example

- Router detects a better route available
- Allows host to have small routing table
**ICMP Echo And ICMP Echo Reply**

- Echo request/reply (ping)
  - Test if destination is reachable/responding
- Request contains an optional data area, identifier (process id), and sequence number
- Reply contains a copy of the request data area, identifier, and sequence number

**Traceroute Example**

trace route to yahoo.com (204.71.177.35), 30 hops max, 40 byte packets
1 gateway.cs.wustl.edu (128.252.165.249) 1.573 ms 0.985 ms 0.986 ms
2 wustl-fddi-starnet.wustl.edu (128.252.5.254) 2.459 ms 2.045 ms 2.184 ms
3 fe0-0.starnet1.starnet.net (199.217.254.194) 2.747 ms 2.223 ms 1.563 ms
4 vcp.stl1.verio.net (129.250.16.97) 2.906 ms 2.243 ms 3.179 ms
5 stl1.stl0.verio.net (129.250.2.213) 3.080 ms 2.736 ms 2.990 ms
6 stl0.dfw2.verio.net (129.250.2.217) 20.986 ms 19.754 ms 20.199 ms
7 dfw2.iad3.verio.net (129.250.2.210) 65.729 ms 63.791 ms 64.099 ms
8 iad3.iad0.verio.net (129.250.2.177) 64.419 ms 64.609 ms 63.755 ms
... 26 pos1-0-622M.cr1.NUQ.globalcenter.net (206.251.0.73) 141.579 ms 148.512 ms 137.012 ms
27 pos5-0-0-155M.hr1.NUQ.globalcenter.net (206.251.0.121) 141.579 ms 148.512 ms 137.012 ms
... 28 yahoo.com (204.71.177.35) 129.703 ms 126.576 ms 137.004 ms

**Traceroute**

- Uses UDP, ICMP and TTL field in IP header
  - Recommended TTL = 64, but some set as high as 255
- Each router along path decrements TTL by 1 or number of seconds it holds datagram
- TTL prevents infinite loops
- When TTL = 0, router returns ICMP "time exceeded" error and router IP address to source

**Traceroute Operation**

- Send UDP datagram to unlikely port at dest with TTL = 1, 2, 3, ...
- Discover routers along path as ICMP message returns

**Beware**

- 1) Routes can change; 2) ICMP packet route may be different than UDP packet; 3) ICMP message contains source IP address of interface at arrival (record route uses interface at departure)