IP – Fragmentation and Reassembly

- IP datagrams can be up to 65,535 bytes – much larger than most networks can transmit in one packet
- Each network type defines *maximum transmission unit* (MTU) – maximum number of bytes that can be carried in payload of link-level frame
Fragmentation (cont.)

- Originating host chooses size for datagram – MTU of host’s network is a good choice
  - Then fragmentation is only required if an intermediate network has a smaller MTU
  - If originating host sends a datagram larger than the network MTU, source host must fragment in IP layer
• If fragmentation required, the `ident`, `flags`, and `offset` fields in header are used
  – 16-bit `ident` the same for every fragment of datagram; chosen by source host, should be unique for each datagram sent within some reasonable time period
  – Destination host assembles all fragments with same `ident`; if some do not arrive, all others are discarded. IP does not attempt to recover missing fragments
Fragmentation (cont.)

- **flags** contains an M bit that indicates that there are more fragments to follow

- **offset** is number of 8-byte chunks that have been sent before this fragment (fragmentation must be done on 8-byte boundaries)

- Fragments are reassembled at destination host, not each intermediate
Datagram Forwarding in IP

• Datagram forwarding
  – Every packet includes dest. IP address
  – Network portion of IP address identifies a single unique network on the Internet
  – Every interface attached to a physical network must have the same network portion of its IP address
  – Every network must be connected to at least one router, which is connected to at least one other network
Forwarding (cont.)

- Check network part of destination address with network part of address for each interface
  - If match, deliver packet directly on that interface
  - Otherwise, need to send packet to next hop router
    - Forwarding table contains (network #, next router) pairs
    - Node also has default router
Address Resolution Protocol (ARP)

• Protocol used to determine a physical address given the host portion of an IP address

• Dynamically learns address mappings to build lookup table – *ARP cache* or *ARP table*

• Entries in ARP table have TTL; they are removed periodically, to handle dynamic changes to network (every 15 min)
ARP (cont.)

- ARP invoked whenever no match found in table
- Broadcast ARP query with target IP addr.
- Any host with match responds
- Sender extracts link-level address from reply and adds to table
- ARP query also includes senders IP and link-level addresses, so everyone on network can update its own ARP table
ARP (cont.)

• If address is already in ARP table, just refresh entry (reset TTL)
• If receiver is target of query, always adds sender
• Other nodes that hear query don’t add new entry
ARP Packet Format

- Mapping for IP – Ethernet

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware type = 1</td>
<td>Protocol Type = 0x0800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLen = 48</td>
<td>PLen = 32</td>
<td>Operation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source hardware address (bytes 0 – 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source HW Addr (bytes 4 – 5)</td>
<td>Source Protocol Addr (bytes 0 – 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Protocol Addr (bytes 2 – 3)</td>
<td>Dest HW Addr (bytes 0 – 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dest HW Addr (bytes 2 – 6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dest Protocol Addr (bytes 0 – 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP Addresses (cont.)

- HLen is hardware address length, PLen is protocol address length, operation indicates request vs. response
- Might add ARP info to regular forwarding table
Dynamic Host Configuration Protocol (DHCP)

• Used to simplify task of configuring hosts on a network with the necessary IP address and default router

• Automatic operation based on broadcast messages

• Each network should have at least one DHCP server
DHCP (cont.)

- Server has default router information, and either a pre-configured table of hosts, indexed by link-level address, or a pool of available IP addresses that are handed out on demand
  - Dynamically allocated addresses are only *leased*; host must renew lease when it expires
- Following example of `ipconfig` on Windows
DHCP (cont.)

- Windows IP Configuration

  Host Name: Odin
  Primary Dns Suffix: 
  Node Type: Unknown
  IP Routing Enabled: No
  WINS Proxy Enabled: No

  Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix: 
  Description: Intel(R) PRO/1000 CT Desktop Connection
  Physical Address: 00-50-2C-0A-0B-EE
  Dhcp Enabled: Yes
  Autoconfiguration Enabled: Yes
  IP Address: 192.168.0.3
  Subnet Mask: 255.255.255.0
  Default Gateway: 192.168.0.1
  DHCP Server: 192.168.0.1
  DNS Servers: 69.51.76.26
               69.51.76.36
  Lease Obtained: Tuesday, October 25, 2005 10:46:20 PM
  Lease Expires: Friday, October 28, 2005 10:46:20 PM
DHCP (cont.)

- Newly booted client broadcasts DHCPDISCOVER message; if DHCP server present, responds with config.
- If no DHCP server on network, one node can act as relay agent
  - Relay agent knows IP address of DHCP server, and just resends discovery request to server, and relays response back to original client
DHCP (cont.)

• Based on older BOOTP protocol (so some fields in DHCP packet no longer apply).
Internet Control Message Protocol (ICMP)

- Defines error messages to be returned to sender if router cannot process datagram successfully
  - Destination network or host unreachable
  - Reassembly process failed (fragment lost)
  - TTL reached 0
  - IP header checksum failed
  - Source Quench to control congestion
  - Echo to send round-trip between nodes
ICMP (cont.)

• Additional control messages
  – ICMP-Redirect tells source to update routing table

• *Ping* uses the echo message and *traceroute* uses the TTL message to measure performance through network
ICMP Ping

- C:\>ping www.google.com

Pinging www.l.google.com [66.102.7.99] with 32 bytes of data:

Reply from 66.102.7.99: bytes=32 time=184ms TTL=239
Reply from 66.102.7.99: bytes=32 time=544ms TTL=239
Reply from 66.102.7.99: bytes=32 time=798ms TTL=239
Reply from 66.102.7.99: bytes=32 time=693ms TTL=239

Ping statistics for 66.102.7.99:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 184ms, Maximum = 798ms, Average = 554ms
ICMP Traceroute

- C: \> tracert www.google.com

  Tracing route to www.l.google.com [66.102.7.104] over a maximum of 30 hops:

  1  <1 ms  <1 ms  <1 ms  192.168.0.1
  2   2 ms   2 ms   2 ms   64.25.129.146
  3  548 ms  501 ms  359 ms  64.25.129.145
  4   14 ms  58 ms   59 ms   64.25.130.17
  5   15 ms  50 ms   74 ms   69.51.77.157
  6  505 ms  222 ms  109 ms  ge-0-1-0-fh-sea.mt.core.transaria.net [69.51.76.46]
  7   80 ms   50 ms  127 ms  ge-0-1-0-fh-sea.mt.core.transaria.net [69.51.76.46]
  8  333 ms  328 ms  760 ms    12.118.34.5
  9 1010 ms  905 ms 1034 ms   12.122.80.234
 10  326 ms  137 ms  70 ms   ggr1-p340.st6wa.ip.att.net [12.123.44.129]
 11  369 ms  569 ms  502 ms  so-3-2-0.gar1.Seattle1.Level3.net [4.68.127.109]
 12  468 ms  546 ms  255 ms  ae-1-55.mp1.Seattle1.Level3.net [4.68.105.129]
 13  167 ms  125 ms  76 ms   as-0-0.bbr2.SanJose1.Level3.net [64.159.0.218]
 14   64 ms   65 ms   59 ms   ae-22-56.car2.SanJose1.Level3.net [4.68.123.176]
 15   95 ms   95 ms  117 ms  unknown.Level3.net [209.247.202.218]
 16  216 ms  146 ms  105 ms   66.249.94.31
 17  393 ms  312 ms  472 ms  216.239.49.146
 18  512 ms  537 ms  790 ms   66.102.7.104

  Trace complete.
Virtual Private Network (VPN)

- Share a large public network (i.e. the Internet), but create a virtual subset of the network that is only accessible by one organization
  - Keep data from hosts outside the VPN out
  - Keep data from hosts inside the VPN secure
- One possibility – use virtual circuits, administratively control who can establish them
VPN (cont.)

• Create *IP tunnels* – virtual point-to-point links over a public switched network
  – Encapsulate an IP packet to a host on the destination network inside an IP packet to the router at the other end of tunnel
  – Implement virtual interface to handle adding the extra headers before sending out over physical interface
  – Can also encrypt payload to add security
VPN (cont.)

- Can encapsulate non-IP packets to send through IP network
- Can also use to redirect IP packet to a network other than the one its header indicates; used for mobile host forwarding