Internet Control Protocols
Reading: Chapter 3

ARP - RFC 826, STD 37
DHCP - RFC 2131
ICMP - RFC 0792, STD 05
Goals of Today’s Lecture

• Bootstrapping an end host
  – Learning its own configuration parameters (DHCP)
• IP level issues
  – Error reporting and monitoring (with ICMP)
• Locating Host
  – Learning the link-layer addresses of other nodes (ARP)
Thus Far in the Class…

- HTML
- TCP
- IP
- Ethernet

HTTP message

TCP segment

- router
- router

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

- host
- host

CS 125 - myarpicmp
Thus Far in the Class…

• Application protocols
  – SNMP - Simple Network Management Protocol

• Mentioned: Transport services built on IP
  – TCP: reliable byte stream with congestion control
  – UDP: unreliable message delivery

• Internet Protocol (IP)
  – Best-effort packet delivery service
  – IP addresses and IP prefixes
  – Packet forwarding based on longest-prefix match

• NOW
  – Try to set up communication via Addresses
  – Try to investigate IP traffic
How To Bootstrap an End Host?

• What IP address the host should use? Why?
• What local Domain Name System server to use? Why?
• How to send packets to remote destinations?
• How to ensure incoming packets arrive? But best effort
Avoiding Manual Configuration

- Dynamic Host Configuration Protocol (DHCP)
  - End host learns how to send packets
  - Learn IP address, DNS servers, and gateway
- Address Resolution Protocol (ARP)
  - Others learn how to send packets to the end host
  - Learn mapping between IP address and MAC address
  - Why? …All traffic is embedded in media (MAC) frames and delivered to host that way
Key Ideas in Both DHCP & ARP

• **Broadcasting:** when in doubt, shout!
  – Broadcast query to all hosts in the local-area-network
  – … when you don’t know how to identify the right one

• **Caching:** remember the past for a while
  – Store the information you learn to reduce overhead
  – Remember your own address & other host’s addresses

• **Soft state:** eventually forget the past
  – Associate a time-to-live field with the information
  – … and either refresh or discard the information
  – Key for robustness in the face of unpredictable change
LANS:
Need Yet Another Kind of Identity

- LANs are designed for arbitrary network protocols
  - Not just for IP and the Internet
- Using IP address would require reconfiguration
  - Every time the adapter was moved or powered up
- Broadcasting all data to all adapters is expensive
  - Requires every host on the LAN to inspect each packet
- Do NOT know the media type

Motivates separate Medium Access Control (MAC) addresses
  -- Media Level
Also motivated by IP riding on any media
MAC Address vs. IP Address

- MAC addresses
  - Hard-coded (not really) in read-only memory when adaptor is built
  - Like a social security number
  - Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76) moving to 64
  - Portable, and can stay the same as the host moves
  - Used to get packet between interfaces on same network

- IP addresses
  - Configured, or learned dynamically
  - Like a postal mailing address
  - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
  - Not portable, and depends on where the host is attached
  - Used to get a packet to destination IP subnet
MAC Addresses on a LAN

- 1A-2F-BB-76-09-AD
- 58-23-D7-FA-20-B0
- 0C-C4-11-6F-E3-98
- 71-65-F7-2B-08-53

LAN

= adapter
Bootstrapping Problem

- Host doesn’t have an IP address yet
  - So, host doesn’t know what source IP address to use
- Host doesn’t know who to ask for an IP address
  - So, host doesn’t know what destination address to use
- Solution: shout to discover a server who can help
  - Broadcast a server-discovery message
  - Server sends a reply offering an address
Broadcasting

- Broadcasting: sending to everyone
  - Special destination address: FF-FF-FF-FF-FF-FF
  - All adapters on the LAN receive the packet

- Delivering a broadcast packet
  - Easy on a “shared media”
  - Like shouting in a room – everyone can hear you
  - E.g., Ethernet, wireless, and satellite links
  - Switches MUST forward

![Diagram showing broadcasting in shared wire (e.g. Ethernet), shared wireless (e.g. Wavelan), satellite, and cocktail party]
Response from the DHCP Server

- DHCP “offer message” from the server
  - Configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (the time the information remains valid)
- Multiple servers may respond
  - Multiple servers on the same broadcast media
  - Each may respond with an offer
  - The client can decide which offer to accept
- Accepting one of the offers
  - Client sends a DHCP request echoing the parameters
  - The DHCP server responds with an ACK to confirm
  - … and the other servers see they were not chosen
Dynamic Host Configuration Protocol

arriving client

DHCP discover (broadcast)

DHCP offer

DHCP request (broadcast)

DHCP ACK

DHCP server 233.1.2.5

I accept
• DHCP in Operation

![Diagram showing DHCP in Operation]

- Host
  - Broadcast
    - DHCP relay
    - Other networks
      - Unicast to server
        - DHCP server
Chapter 4, Figure

- Broadcast
- DHCP Discover: NSC
- IP Address: 255.255.255.255
- Broadcast
- DHCP relay
- Other networks
- Unicast to server
- Unicast
- Unicast
- Reply
- One per network
- Table: GOD!
### DHCP Packet Format

<table>
<thead>
<tr>
<th>Operation</th>
<th>HType</th>
<th>HLen</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secs</td>
<td></td>
<td>Flags</td>
<td></td>
</tr>
<tr>
<td>ciaddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yiaddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>siaddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>giaddr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chaddr (16 bytes)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sname (64 bytes)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file (128 bytes)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Request/Reply

<table>
<thead>
<tr>
<th>Operation</th>
<th>HType</th>
<th>HLen</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION ID</td>
<td></td>
<td>Xid</td>
<td>MATCH ADDR/RES.</td>
</tr>
<tr>
<td>Secs</td>
<td></td>
<td></td>
<td>Flags</td>
</tr>
</tbody>
</table>

- ciaddr: CLIENT
- yiaddr: ASSIGN NEW
- siaddr: Server IP
- giaddr: Gateway IP

- chaddr (16 bytes)
- sname (64 bytes)
- file (128 bytes)

Options

Chapter 4, Figure
Deciding What IP Address to Offer

- Server as centralized configuration database
  - All parameters are statically configured in the server
  - E.g., a dedicated IP address for each MAC address
  - Avoids complexity of configuring hosts directly
  - … while still having a permanent IP address per host
- Or, dynamic assignment of IP addresses
  - Server maintains a pool of available addresses
  - … and assigns them to hosts on demand
  - Leads to less configuration complexity
  - … and more efficient use of the pool of addresses
  - Though, it is harder to track the same host over time
- Or Mix of the two
Soft State: Refresh or Forget

- Why is a lease time necessary?
  - Client can release the IP address (DHCP RELEASE)
    - E.g., “ipconfig /release” at the DOS prompt
    - E.g., clean shutdown of the computer
  - But, the client might not release the address
    - E.g., the host crashes (blue screen of death!)
    - E.g., buggy client software
  - And you don’t want the address to be allocated forever

- Performance trade-offs
  - Short lease time: returns inactive addresses quickly
  - Long lease time: avoids overhead of frequent renewals
So, Now the Host Knows Things

- IP address
- Mask
- Gateway router
- DNS server
- …

Get from a DHCP server that knows all

- And can send packets to other IP addresses
  - But, how to learn the MAC address of the destination?
Sending Packets Over a Link

- Adaptors only understand MAC addresses
  - Translate the destination IP address to MAC address
  - Encapsulate the IP packet inside a link-level frame
Address Translation

• How, given an IP address does a host send a datagram to a physical address?
• Map IP addresses into physical addresses
  – destination host, or
  – next hop router
• Techniques
  – encode physical address in host part of IP address (IPv6)
  – table-based, need to build and maintain table
• ARP Address Resolution Protocol
  – returns physical address
  – table of IP to physical address bindings
  – broadcast request if IP address not in table
  – target machine responds with its physical address
  – table entries are discarded if not refreshed
ARP - Address Resolution Protocol Table

• Every node maintains an ARP table, ARP table at each host!!
  – (IP address, MAC address) pair
• Consult the table when sending a packet
  – Map destination IP address to destination MAC address
  – Encapsulate and transmit the data packet
• But, what if the IP address is not in the table?
  – Sender broadcasts: “Who has IP address 1.2.3.156?”
  – Receiver responds: “MAC address 58-23-D7-FA-20-B0”
  – Sender caches the result in its ARP table
• No need for network administrator to get involved
Figure 4.2 Operation of ARP when user types "ftp hostname".
Example: A Sending a Packet to B

How does host A send an IP packet to host B?

A sends packet to R, and R sends packet to B.
Host A Decides to Send Through R

- Host A constructs an IP packet to send to B
  - Source 111.111.111.111, destination 222.222.222.222

- Host A has a gateway router R
  - Used to reach destinations outside of 111.111.111.0/24
  - Address 111.111.111.110 for R learned via DHCP
Host A Sends Packet Through R

- Host A learns the MAC address of R’s interface
  - ARP request: broadcast request for 111.111.111.110
  - ARP response: R responds with E6-E9-00-17-BB-4B

- Host A encapsulates the packet and sends to R
R Decides how to Forward Packet

- Router R’s adaptor receives the packet
  - R extracts the IP packet from the Ethernet frame
  - R sees the IP packet is destined to 222.222.222.222
- Router R consults its forwarding table
  - Packet matches 222.222.222.0/24 via other adaptor
R Sends Packet to B

- Router R’s learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-56-2A
- Router R encapsulates the packet and sends to B
ARP Details

• Request Format
  – HardwareType: type of physical network (e.g., Ethernet)
  – ProtocolType: type of higher layer protocol (e.g., IP)
  – HLEN & PLEN: length of physical and protocol addresses
  – Operation: request or response
  – Source/Target-Physical/Protocol addresses

• Notes
  – table entries timeout in about 10 minutes
  – update table with source when you are the target
  – update table if already have an entry
  – do not refresh table entries upon reference
## ARP Packet Format

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware type = 1</td>
<td>ProtocolType = 0x0800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLen = 48</td>
<td>PLen = 32</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 0 — 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 4—5)</td>
<td>SourceProtocolAddr (bytes 0—1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceProtocolAddr (bytes 2 — 3)</td>
<td>TargetHardwareAddr (bytes 0—1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetHardwareAddr (bytes 2—5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetProtocolAddr (bytes 0 — 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
/ * arp.h - SHA, SPA, THA, TPA */

/* Internet Address Resolution Protocol (see RFCs 826, 920) */
#define AR_HARDWARE  1 /* Ethernet hardware type code */

/* Definitions of codes used in operation field of ARP packet */
#define AR_REQUEST  1 /* ARP request to resolve address */
#define AR_REPLY  2 /* reply to a resolve request */
#define RA_REQUEST  3 /* reverse ARP request (RARP packets) */
#define RA_REPLY  4 /* reply to a reverse request (RARP *) */

struct arp {
    short ar_hatype;  /* hardware type */
    short ar_ptype;  /* protocol type */
    char ar_halen;  /* hardware address length */
    char ar_phlen;  /* protocol address length */
    short ar_op;  /* ARP operation (see list above) */
    char ar_spa[ ];  /* sender's physical hardware address */
    char ar_sha[ ];  /* sender's protocol address (IP addr.) */
    char ar_tha[ ];  /* target's physical hardware address */
    char ar_tpa[ ];  /* target's protocol address (IP) */
};

#define SHA(p) (&p->ar_addr[0])
#define SPA(p) (&p->ar_addr[p->ar_halen])
#define THA(p) (&p->ar_addr[p->ar_halen + p->ar_pllen])
#define TPA(p) (&p->ar_addr[(p->ar_halen*2) + p->ar_pllen])
struct arpentry {
    short ae_state;  /* format of entry in ARP cache */
    short ae_hwtype; /* state of this entry (see below) */
    short ae_ptype;  /* hardware type */
    char ae_hlen;    /* protocol type */
    char ae_prlen;   /* hardware address length */
    struct netif *ae_pni; /* protocol address length */
    int ae_queue;    /* pointer to interface structure */
    int ae_attempts; /* queue of packets for this address */
    int ae_ttl;      /* number of retries so far */
    char ae_hwa[MAXHWLEN]; /* time to live */
    /* Hardware address */
    char ae_pag[MAXPRALEN]; /* Protocol address */
};

#define AS_FREE    0 /* Entry is unused (initial value) */
#define AS_PENDING 1 /* Entry is used but incomplete */
#define AS_RESOLVED 2 /* Entry has been resolved */

/* RARP variables */
extern int rarppid; /* id of process waiting for RARP reply */
extern int rarpsmem; /* semaphore for access to RARP service */

/* ARP variables */
extern struct arpentry arptable[ARP_TSIZE];
Error Reporting & Other IP Issues

• Examples of errors a router may see
  – Router doesn’t know where to forward a packet
  – Packet’s time-to-live field expires

• Router doesn’t really need to respond
  – Best effort means never having to say you’re sorry
  – So, IP could conceivably just silently drop packets

• But, silent failures are really hard to diagnose
  – IP includes basic feedback about network problems
  – Internet Control Message Protocol (ICMP)
Internet Control Message Protocol (ICMP)

- Sent to Source Host - NOT Routers (why?)
- Echo (ping)
- Redirect (from router to source host)
- Destination unreachable (protocol, port, or host)
- TTL exceeded (so datagrams don’t cycle forever)
- Checksum failed
- Reassembly failed
- Cannot fragment
Internet Control Message Protocol

- ICMP runs on top of IP
  - In parallel to TCP and UDP
  - Though still viewed as an integral part of IP

- Diagnostics
  - Triggered when an IP packet encounters a problem
    - E.g., time exceeded or destination unreachable
  - ICMP packet sent back to the source IP address
    - Includes the error information (e.g., type and code)
    - … and an excerpt of the original data packet for identification
  - Source host receives the ICMP packet
    - And inspects the excerpt of the packet (e.g., protocol and ports)
    - … to identify which socket should receive the error
IP Error Handling

What’s a gateway to do?
Gateway/Router is autonomous
  - No real coordination with source
  - Destination machine could be off
  - Time-to-Live gone
  - Congestion at Router/Gateway
Use Internet Control Message Protocol - ICMP
  “Any IP device to any IP device”
Rides in IP
Example: Time Exceeded

- Host sends an IP packet
  - Each router decrements the time-to-live field
- If time-to-live field reaches 0
  - Router generates an ICMP message
  - Sends a “time exceeded” message back to the source
Traceroute: Exploiting “Time Exceeded”

- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of \( n \)
  - Each router along the path decrements the TTL
  - “TTL exceeded” sent when TTL reaches 0
- Traceroute tool exploits this TTL behavior

Send packets with TTL=1, 2, … and record source of “time exceeded” message
Ping: Echo and Reply

• ICMP includes a simple “echo” function
  – Sending node sends an ICMP “echo” message
  – Receiving node sends an ICMP “echo reply”
• Ping tool
  – Tests the connectivity with a remote host
  – … by sending regularly spaced echo commands
  – … and measuring the delay until receiving the reply
• Pinging a host
  – “ping www.cs.princeton.edu” or “ping 12.157.34.212”
  – Used to test if a machine is reachable and alive
  – (However, some nodes have ICMP disabled… 😞)
<table>
<thead>
<tr>
<th>type</th>
<th>code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (Ping reply, Chapter 7)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>destination unreachable:</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>network unreachable (Section 9.3)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>host unreachable (Section 9.3)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>protocol unreachable</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>port unreachable (Section 6.5)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>fragmentation needed but don't-fragment bit set (Section 11.6)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>source route failed (Section 8.5)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>destination network unknown</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>destination host unknown</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>source host isolated (obsolete)</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>destination network administratively prohibited</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>destination host administratively prohibited</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>network unreachable for TOS (Section 9.3)</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>host unreachable for TOS (Section 9.3)</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>communication administratively prohibited by filtering</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>host precedence violation</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>precedence cutoff in effect</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (elementary flow control, Section 11.11)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>redirect (Section 9.5):</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>redirect for network</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>redirect for host</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>redirect for type-of-service and network</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>redirect for type-of-service and host</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (Ping request, Chapter 7)</td>
</tr>
</tbody>
</table>
Conclusion

• Important control functions
  – Bootstrapping
  – Error reporting and monitoring

• Internet control protocols
  – Dynamic Host Configuration Protocol (DHCP)
  – Address Resolution Protocol (ARP)
  – Internet Control Message Protocol (ICMP)