BOOTP, DHCP and NAT

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Overview

- Bootstrapping (Diskless workstations)
- BOOTP
- Dynamic address allocation
- DHCP
- Private Addresses: NAT, RSIP
- Ref: Chap 16, Doug Comer’s TCP/IP book,
  IETF NAT Working Group

Bootstrapping

- Computer loads a simple boot program. The boot program loads operating system.
- On diskless machine, the computer needs to know the network address of the o/s file
- It needs to know its own IP address, subnet mask, IP address of default router, IP address of DNS server
- It only knows its h/w address.

Configuration

- Different nodes have different parameters
- Configuration = Setting the parameters
- Key parameters for IP hosts:
  - IP Address
  - Default router address
  - Subnet mask
  - Name
  - DNS server IP addresses

Key RARP Limitations

- RARP: user process over link layer directly
  - RARP server system-dependent
  - Needs to interface with link layer driver directly => separate filters and direct access to hardware needed
  - Returns only IP address
  - Booting and configuration params not returned even though there is space in packet
  - Host needs ICMP and TFTP to complete booting
- Can’t relay RARP requests to a central server.
- Need RARP server per broadcast domain

Method 2: BOOTP

- Runs over UDP/IP!
  - IP software can broadcast (to 255.255.255.255) even if local IP address unknown => client broadcasts BOOTP request
  - Port number 67 for server and 68 for client (not an ephemeral port)
  - Delivers BOOTP reply to BOOTP client and not other UDP apps when reply is broadcast
  - Does not wake up other servers during broadcast reply
**BOOTP (Continued)**

- BOOTP requests/replies sent w/ DF bit set.
- Server can send reply via broadcast or unicast:
  - For unicast reply, BOOTP server knows the IP address, but the link layer address is not in the ARP cache
  - Note that the server cannot send an ARP message because client does not know its IP address
  - Server can use ioctl(8) (or arp -s) to set the value of the cache based upon BOOTP request => can do this only if it has permission

**BOOTP Features (Continued)**

- Else send broadcast reply
- Reply: IP Address, Boot Server IP address, Default Router, Boot file name, subnet mask
- More information, but still only a single packet exchange
- Client gets boot image using TFTP => booting still a 2-step process

**BOOTP features (Continued)**

- Advantages of using UDP/IP:
  - Bootstrapping can occur across a router via a relaying mechanism
  - BOOTP uses checksum provided by UDP
  - Multiple requests/replies
    - Process the first one
    - Client uses a transaction ID field to sort out replies
  - Clients responsible for reliability
    - Uses timeout, retransmission & exponential backoff
    - Random initial timeout (btwn 0 & 4s): simultaneous reboot after power restoration

**BOOTP Message Format**

<table>
<thead>
<tr>
<th>Operation Code</th>
<th>H/W Type</th>
<th>H/W Length</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BOOTP Message (Continued)**

- Operation: 1 = Request, 2 = Reply
- H/w type: 1 = Ethernet
- H/w Address Length
- Hops: Initialized to zero. Incremented by BOOTP relays (routers)

**BOOTP Message**

- Boot File name: Generic name like "unix" in the request. Full name in response.
- Vendor specific area: Misnomer. Also used for general purpose info.
- Magic cookie: First 4 octets = 99.130.83.99
- Type-length-value: describes the option

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Time of Day</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>End</td>
<td>255</td>
<td>-</td>
</tr>
</tbody>
</table>
DHCP

- BOOTP limitation: cannot dynamically assign IP address
- Dynamic Host Configuration Protocol (DHCP)
  - BOOTP + Dynamic allocation of IP addresses => compatible with BOOTP.
  - No new fields in header.
  - Addresses can be leased for a period. Reallocated to the same or other nodes after lease expiry.

DHCP Message Format

- Slightly modified version of BOOTP message =>
  - A DHCP server can be programmed to answer BOOTP requests
- BOOTP's Unused field renamed to "Flags"
- Only one bit of 16-bit Flags has been defined
  - Left-most flag bit = 1 => Servers, please reply using IP broadcast address
  - Servers by default send hardware unicast response
- Vendor-specific field renamed to "Options"
  - Size increased to 312 bytes (from 64 bytes)
  - Option type 53 specifies the "type of the message"

DHCP (Continued)

- "Option overload"
  - Server Host name and boot file name when unused for their original purpose could be used to code more options

DHCP State Diagram

- Host Boots => INITIALIZE state
- DHCPDISCOVER: broadcast request to servers => SELECT state
- DHCPOFFER (from server) => remain in SELECT
- DHCPREQUEST => select one of the offers and notify server (goto REQUEST state) about the lease
DHCP States (Continued)
- DHCPACK => server ok request to lease => go to the BOUND state
- Renewal: after 50% of lease go to RENEW state
- Rebind: after 87.5% of time, if server has not responded, try again and go to REBIND.
- If server NACKs or lease expires, or client sends DCHPRELEASE, go to INITIALIZE, else come back to BOUND state

Private vs Public Addresses
- Since IPv4 addresses are scarce, enterprises may use private addresses within their “realms”

<table>
<thead>
<tr>
<th>Class</th>
<th>Private Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.0.0.0 ... 10.255.255</td>
</tr>
<tr>
<td>B</td>
<td>172.16.0.0 ... 172.16.255.255</td>
</tr>
<tr>
<td>C</td>
<td>192.168.0.0 ... 192.168.255.255</td>
</tr>
</tbody>
</table>

- Need to get “globally unique” public addresses for external use.
- Mapping between public & private addresses done by NAT (Network Address Translator)

Simple NAT operation

Dynamic NAT = NAT + DHCP

Network Address Port Translation (NAPT)

NAPT (contd)
- Also known as IP masquerading. Allows many hosts to share a single IP address differentiated by port numbers.
- Eg: Suppose private hosts 192.168.0.2 and 192.168.0.3 send packets from source port 1108.
- NAPT translates these to a single public IP address 206.245.160.1 and two different source ports, say 61001 and 61002.
- Response traffic received for port 61001 is routed back to 192.168.0.2:1108.
- Traffic for port 61002 traffic is routed back to 192.168.0.3:1108.
Realm-Specific IP (RSIP)

- NAT (and NAPT) have to mess with several transport/application level fields.
- NAT breaks IPSec. Solution: RSIP
- RSIP leases public IP addresses and ports to RSIP hosts => not transparent like NAT.
- RSIP does not operate in stealth mode and does not translate addresses on the fly.
- RSIP allows hosts to directly participate concurrently in several addressing realms.
- Avoids violating the end-to-end nature of the Internet => allows IPSec

Summary

- RARP allows finding an IP address
- BOOTP allows default router, subnet mask, DNS
- DHCP allows dynamic allocation
- DHCP is backward compatible with BOOTP
- NAT, NAPT, RSIP allow use of private addresses and smaller pool of public addresses