RARP, BOOTP DHCP

INTRODUCTION TO RARP, BOOTP AND DHCP, PROTOCOLS FOR DYNAMIC DISTRIBUTION OF NETWORK CONFIGURATION PARAMETERS

Peter R. Egli
INDIGOO.COM

Contents

1. Purpose of RARP/BOOTP/DHCP

Problem:

It is desireable to dynamically allocate IP addresses to hosts / computers for the following reasons.

a. Conserve IP addresses:

IP addresses have become scarce. It is desirable to only assign IP addresses to hosts that are in use.

b. Ease of IP address management:

Statically assigning IP addresses to hosts in a large network is difficult to manage. Distributing IP addresses from a server allows to centrally manage IP address allocation.

c. Mobility of hosts / clients:

With dynamic IP address assignment, hosts can easily move around (mobility) without the need to administratively change the host's IP address.

2. Common IP address assignment protocols

In networks that support multiple access (multiple hosts on the same network such as Ethernet) and broadcast (possibility to send a frame to all hosts on the network) the following protocols have evolved over time for the dynamic assignment of IP addresses.

1. RARP (Reverse ARP):

RARP (defined in RFC903) is an early protocol for dynamic IP address assignment in Ethernet networks.
RARP is severely limited in that it does not support assigning name servers and default gateway. For these reasons RARP has been supplanted by BOOTP and the more modern DHCP.

2. BOOTP (BOOTstrap Protocol):

BOOTP (defined in RFC951) has been introduced for assigning IP addresses and other information such as name server and default gateway.
Additionally BOOTP may be routed between sub-nets since it is based on UDP/IP.

3. DHCP (Dynamic Host Configuration Protocol):

DHCP (defined in RFC2131) eliminates some of the limitations present in BOOTP such as restricted vendor option size and lack of a lease time for IP addresses.

All protocols use (MAC) broadcast request for requesting an IP address.
This is necessary since hosts are unable to communicate on IP level before they are assigned a valid IP address.

3. RARP RFC903 (1/2)

➔ RARP is very simple but not used much anymore (hosts use DHCP instead).

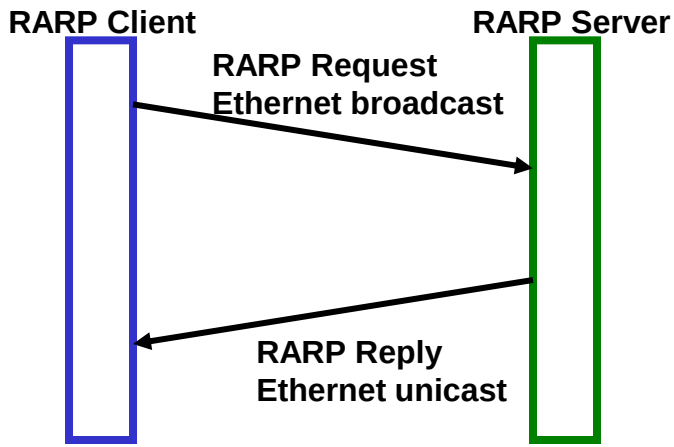
N.B.: RARP, as its name may imply, is not a protocol for querying the IP address based on a MAC address (reverse operation of ARP). RARP is used for assigning an IP address to a host that previously did not have an IP address. The server may use a fixed assignment MAC→IP.

ARP / RARP Packet:

Hardware type = 1		ProtocolType=0x0800
HLen = 48	PLen = 32	Operation
SourceHardwareAddr		
SourceHardwareAddr		SourceProtocolAddr (IP)
SourceProtocolAddr (IP)		TargetHardwareAddr
TargetHardwareAddr		
TargetProtocolAddr (IP)		

Operation:
1 ARP request
2 ARP reply
3 RARP request
4 RARP reply

RARP Transaction:



3. RARP RFC903 (2/2)

RARP purpose:

RARP was defined for booting diskless workstations. These workstations booted with an EEPROM-resident small bootstrap code which acquired an IP address and the name of code file to be downloaded afterwards (via TFTP). The name of the code file was something like „8CFC0D21.SUN4C" where 8CFC0D21 is the hex code of the IP address of the TFTP server and SUN4C is the type of system being bootstrapped (the client assumes that TFTP server = RARP server whose IP is coded in the field *SourceProtocolAddr* in the RARP response).

RARP problems:

1. RARP runs directly on top of Ethernet (no IP layer). Thus RARP can only be used in the same subnet.
2. RARP only allows to assign an IP address but no default gateway (thus the TFTP server with the code file had to be on same subnet) and no name server.
3. RARP is not simple to implement due to limited access to raw Ethernet frames in many systems (Ethernet runs in kernel space).

4. BOOTP BOOTstrap Protocol RFC951 (1/4)

BOOTP was defined to overcome the limitations of RARP.

1. Additional information elements:
In addition to the IP address a default gateway, name server, boot file and many other information elements can be configured (as vendor specific information).

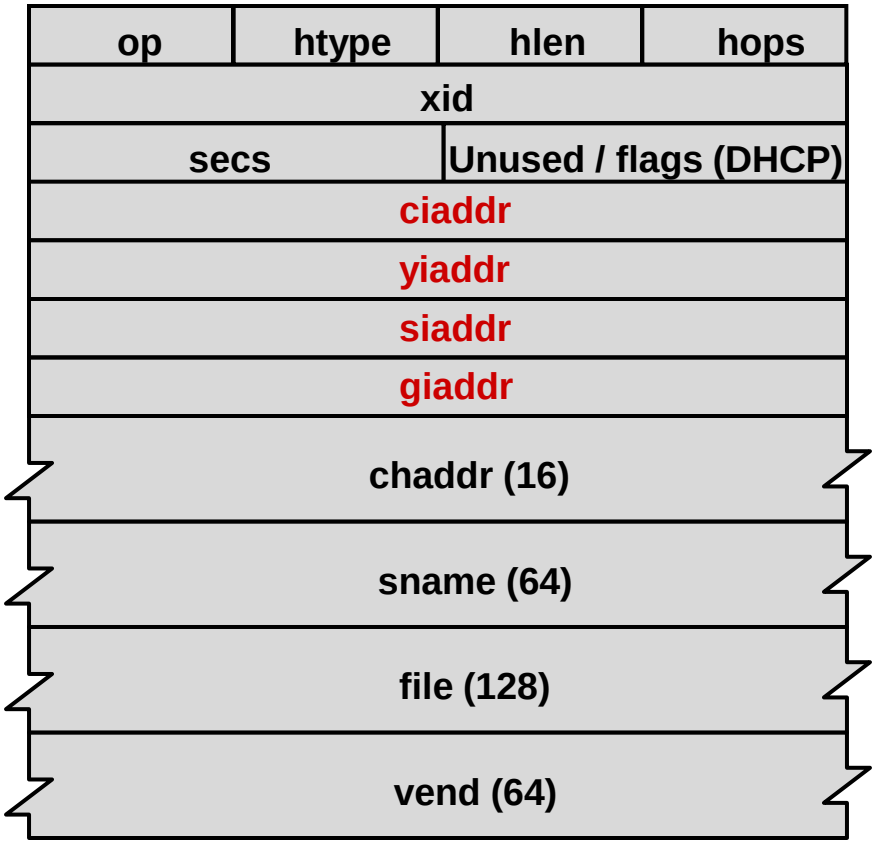
2. Routable:
BOOTP is routable since it uses IP/UDP.
This means that the BOOTP server does not need to be on the same subnet (requires a BOOTP relay agent though).

3. Simple socket application:
A BOOTP server implementation is a simple socket application and thus does not need special system calls to get access to Ethernet frames as RARP does.

4. Extensibility:
BOOTP is extensible in that it allows defining vendor specific options. Vendors can pack proprietary information into such vendor specific extensions. Such vendor options could, for example, carry specific configuration options for the host.

4. BOOTP BOOTstrap Protocol RFC951 (2/4)

BOOTP packet format:



op: Packet op code / message type. 1 = BOOTREQUEST, 2 = BOOTREPLY

htype: Hardware address type, see ARP section in "Assigned Numbers" RFC ('1' = 10mb ethernet).

hlen: Hardware address length (eg '6' for 10mb ethernet).

hops: Client sets to zero, optionally used by gateways in cross-gateway booting. If >=3 packet is usually discarded (thus no BOOTP across more than 3 routers).

xid: Transaction ID, a random number used to match this boot request with the responses it generates.

secs: Seconds elapsed since client started trying to boot (useful if a secondary server is in use). Filled in by client. Unused with BOOTP.

ciaddr: Client IP address. Set to 0.0.0.0 in new request. Set to current client IP in refresh requests.

yiaddr: Your IP address (IP delivered from server to client).

siaddr: Server IP address (BOOTP server IP). In DHCP the meaning of this field changed to "address of the server to use in the next step of the client's bootstrap process".

giaddr: Gateway IP Address (proxy server fills in its own IP address if present).

chaddr: Client hardware address (MAC address). This makes MAC address available to server processes for lookup and matching to IP address (if so configured).

sname: Server hostname. With this name the client can restrict access to specific servers).

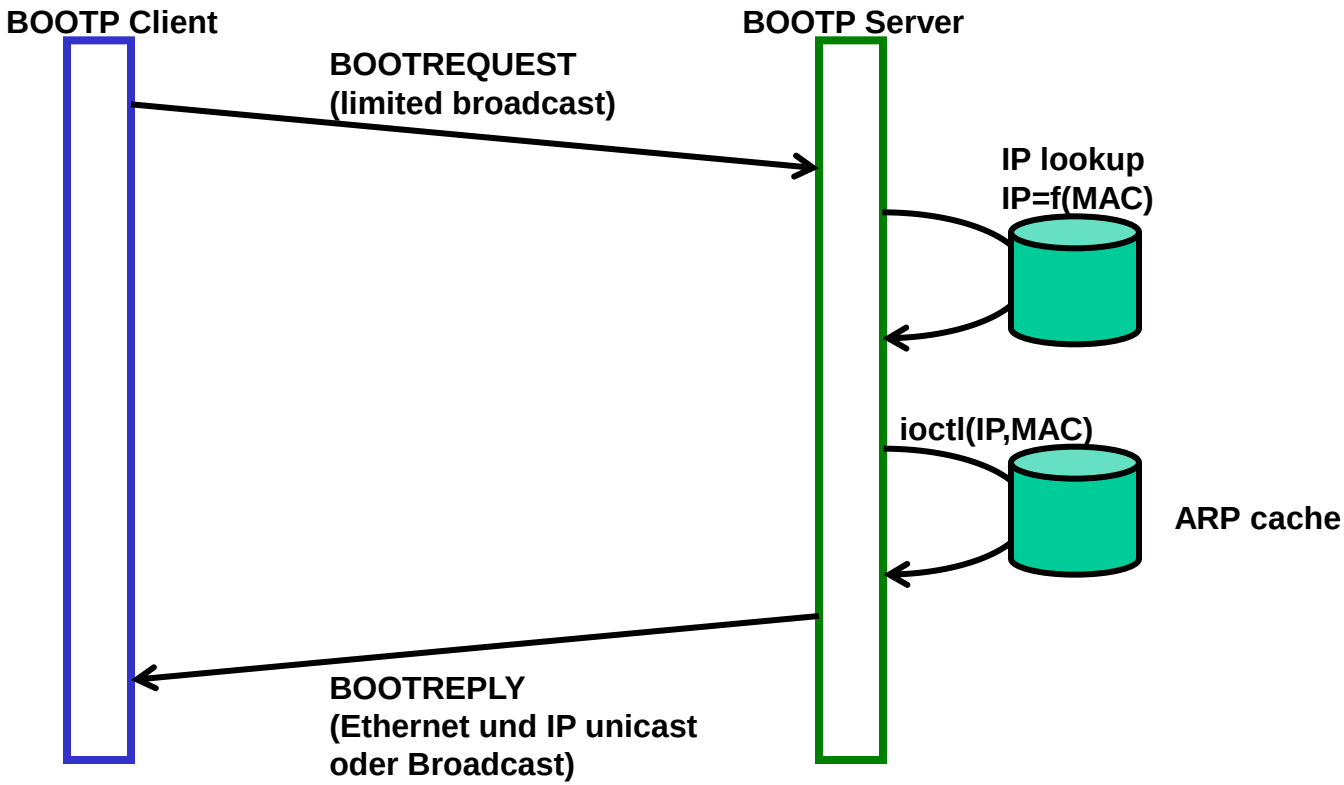
file: Boot filename (for client).

vend: Vendor-specific information (max. 64 bytes, encoding see DHCP) such as default gateway, subnet mask, DNS Server, Time Server, Print Server.

4. BOOTP BOOTstrap Protocol RFC951 (3/4)

BOOTP transaction:

BOOTP has an interesting ‚chicken and egg' problem. Prior to sending the BOOTREPLY, the server must ‚manually' establish an entry in the ARP cache (through a socket `ioctl(IP,MAC)` call). Otherwise the IP layer would not be able to send the frame since there is no mapping IP→MAC in the ARP cache yet (never before a packet with this MAC and the assigned IP address was on the network, thus there is no ARP cache entry).



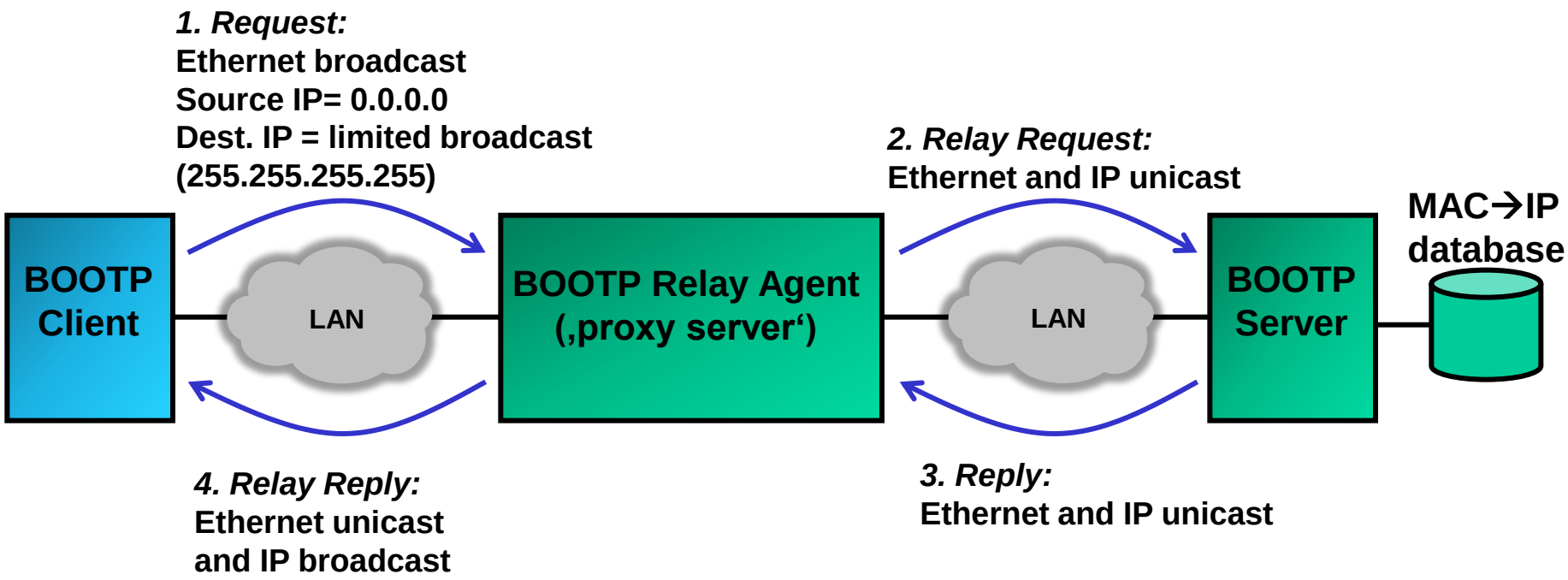
4. BOOTP BOOTstrap Protocol RFC951 (4/4)

BOOTP relay:

With BOOTP relay a single BOOTP server services BOOTP requests from multiple LAN segments. This may be used to ease administration in large networks (operate only one single BOOTP server).

The BOOTP relay agent forwards (unicast) BOOTP requests to a pre-configured BOOTP server.

To reduce broadcasts in the networks, the BOOTP reply is sent back as an Ethernet unicast addressed packet.



5. DHCP Dynamic Host Configuration Protocol RFC2131 (1/5)

Differences between BOOTP and DHCP:

DHCP is an evolution of BOOTP. DHCP was defined to fix the limitations of BOOTP.

1. IP address lease (time):

BOOTP assigns IP addresses without a lease time.
In order to free an IP address assigned by BOOTP, a host needs to be rebooted.
In DHCP an IP address is assigned to a client only for a limited (but extensible) time. If the IP address is not renewed it goes back to the pool of free IP addresses.
Typically a client renews a lease when half of the lease time has elapsed.

2. Vendor options:

In DHCP vendor specific options are no longer restricted to 64 bytes.
This gives more possibilities to carry vendor specific information such as configuration files from DHCP server to client.

5. DHCP Dynamic Host Configuration Protocol RFC2131 (2/5)

BOOTP and DHCP interoperability:

BOOTP and DHCP are interoperable because DHCP is basically an extension of BOOTP.

a. Same ports:

Both BOOTP and DHCP use UDP ports 68 (client) and 67 (server). BOOTP and DHCP use 2 different ports for client and server so that clients do not receive messages to servers and vice versa (DHCP and BOOTP use UDP thus source IP and port number can not be used to distinguish a session or connection).

b. Same packet format:

BOOTP and DHCP use the same packet format. Thus a DHCP server can receive BOOTP packets and vice versa.

c. DHCP transaction on top of BOOTP request-reply:

DHCP uses a 4-way transaction for assigning IP addresses, BOOTP only knows Request and Reply. Therefore the DHCP request packets (client to server) are mapped to BOOTPREQUEST and DHCP server responses are mapped to BOOTPREPLY (every DHCP message is either a BOOTPREQUEST or BOOTPREPLY message).

5. DHCP Dynamic Host Configuration Protocol RFC2131 (3/5)

DHCP message types:

DHCP introduces additional message types to support different scenarios.

DHCPDISCOVER:	**Client broadcast to locate available servers.**
DHCPOFFER:	**Server to client in response to DHCPDISCOVER with an offer containing configuration parameters (offered IP address etc.).**
DHCPREQUEST:	**Client message to servers either (a) requesting offered parameters from one server and implicitly declining offers from all others, (b) confirming correctness of previously allocated address after e.g. a system reboot, or (c) extending the lease on a particular network address.**
DHCPACK:	**Server to client with configuration parameters, including committed IP address.**
DHCPNACK:	**Server to client indicating client's notion of network address is incorrect (e.g. client has moved to new subnet) or client's lease has expired.**
DHCPDECLINE:	**Client to server indicating network address is already in use. DHCP server should then notify the sysadmin.**
DHCPRELEASE:	**Client to server relinquishing IP address and cancelling the remaining lease.**
DHCPINFORM:	**Client to server, asking only for local configuration parameters. Client already has an externally configured network address.**

5. DHCP Dynamic Host Configuration Protocol RFC2131 (4/5)

Trace of DHCP session - client acquires new IP address:

DOS shell commands: `$ipconfig /release; ipconfig /renew`

C: Client → server traffic
S: Server → client traffic

```
C:  1   0.000000   0.0.0.0 -> 255.255.255.255 DHCP DHCP Discover - Transaction ID 0x5e5b0c04
                   Source MAC: 00:10:a4:b2:06:66                  Destination MAC: ff:ff:ff:ff:ff:ff (Broadcast)
                   Source IP: 0.0.0.0 (0.0.0.0)                   Destination IP: 255.255.255.255 (255.255.255.255)
                   ciaddr: 0.0.0.0 (0.0.0.0)                      yiaddr: 0.0.0.0 (0.0.0.0)
                   siaddr: 0.0.0.0 (0.0.0.0)                      giaddr: 0.0.0.0 (0.0.0.0)
                   chaddr: 00:10:a4:b2:06:66                      Option 61 (client identifier): 010010a4b20666 (MAC addr.)
S:  2   1.011502 193.5.54.111 -> 193.5.54.19  DHCP DHCP Offer    - Transaction ID 0x5e5b0c04
                   Source MAC: 08:00:20:c4:a8:c1                  Destination MAC: 00:10:a4:b2:06:66
                   Source IP: 193.5.54.111 (193.5.54.111)         Destination IP: 193.5.54.19 (193.5.54.19)
                   ciaddr: 0.0.0.0 (0.0.0.0)                      yiaddr: 193.5.54.19 (193.5.54.19)
                   siaddr: 193.5.54.111 (193.5.54.111)            giaddr: 0.0.0.0 (0.0.0.0)
                   chaddr: 00:10:a4:b2:06:66                      Option 54 (server identifier): 193.5.54.111
C:  3   1.012289   0.0.0.0 -> 255.255.255.255 DHCP DHCP Request  - Transaction ID 0x5e5b0c04
                   Source MAC: 00:10:a4:b2:06:66                  Destination MAC: ff:ff:ff:ff:ff:ff (Broadcast)
                   Source IP: 0.0.0.0 (0.0.0.0)                   Destination IP: 255.255.255.255 (255.255.255.255)
                   ciaddr: 0.0.0.0 (0.0.0.0)                      yiaddr: 0.0.0.0 (0.0.0.0)
                   siaddr: 0.0.0.0 (0.0.0.0)                      giaddr: 0.0.0.0 (0.0.0.0)
                   chaddr: 00:10:a4:b2:06:66                      Option 54 (server identifier): 193.5.54.111
S:  4   1.032003   193.5.54.111 -> 193.5.54.19  DHCP DHCP ACK     - Transaction ID 0x5e5b0c04
                   Source MAC: 08:00:20:c4:a8:c1                  Destination MAC: 00:10:a4:b2:06:66
                   Source IP: 193.5.54.111 (193.5.54.111)         Destination IP: 193.5.54.19 (193.5.54.19)
                   ciaddr: 0.0.0.0 (0.0.0.0)                      yiaddr: 193.5.54.19 (193.5.54.19)
                   siaddr: 193.5.54.111 (193.5.54.111)            giaddr: 0.0.0.0 (0.0.0.0)
                   chaddr: 00:10:a4:b2:06:66                      Option 54 (server identifier): 193.5.54.111
```

➔ **Broadcast flags in DHCP message: If set to 1 then server must send DHCPOFFER DHCPACK via IP Broadcast (255.255.255.255), because client is not able to receive unicast IP packets (has no IP address yet).**

➔ **'Client identifier' (option 61) is used by the server to identify the client (ascertain the binding MAC – IP address).**
The client uses the 'server identifier' to bind to a specific server (there may be multiple servers responding to the client request).

5. DHCP Dynamic Host Configuration Protocol RFC2131 (5/5)

Trace of DHCP session - client renews current IP address:

DOS shell commands: `$ipconfig /renew`

```
C:  1   0.000000  193.5.54.19 -> 193.5.54.111 DHCP DHCP Request  - Transaction ID 0x3818e428
                  Destination MAC: 08:00:20:c4:a8:c1
                  Source MAC: 00:10:a4:b2:06:66
                  ciaddr: 193.5.54.19 (193.5.54.19)
                  yiaddr: 0.0.0.0 (0.0.0.0)
                  siaddr: 0.0.0.0 (0.0.0.0)
                  giaddr: 0.0.0.0 (0.0.0.0)
                  chaddr: 00:10:a4:b2:06:66
S:  2   0.017734 193.5.54.111 -> 193.5.54.19  DHCP DHCP ACK      - Transaction ID 0x3818e428
                  Destination MAC: 00:10:a4:b2:06:66
                  Source MAC: 08:00:20:c4:a8:c1
                  ciaddr: 193.5.54.19 (193.5.54.19)
                  yiaddr: 193.5.54.19 (193.5.54.19)
                  siaddr: 193.5.54.111 (193.5.54.111)
                  giaddr: 0.0.0.0 (0.0.0.0)
                  chaddr: 00:10:a4:b2:06:66
```

Trace of DHCP session - client releases current IP address:

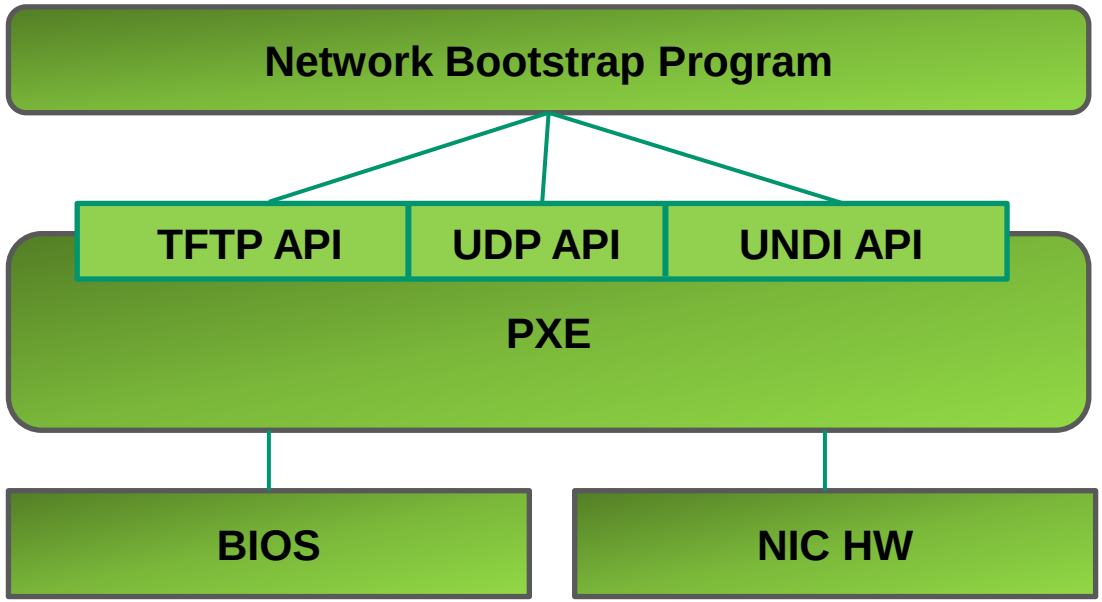
DOS shell command: `$ipconfig /release`

```
C:  1   0.000000  193.5.54.19 -> 193.5.54.111 DHCP DHCP Release  - Transaction ID 0xa04d3a02
                  Destination MAC: 08:00:20:c4:a8:c1
                  Source MAC: 00:10:a4:b2:06:66
                  ciaddr: 193.5.54.19 (193.5.54.19)
                  yiaddr: 0.0.0.0 (0.0.0.0)
                  siaddr: 0.0.0.0 (0.0.0.0)
                  giaddr: 0.0.0.0 (0.0.0.0)
                  chaddr: 00:10:a4:b2:06:66
```

6. PXE – Preboot Execution Environment (1/4)

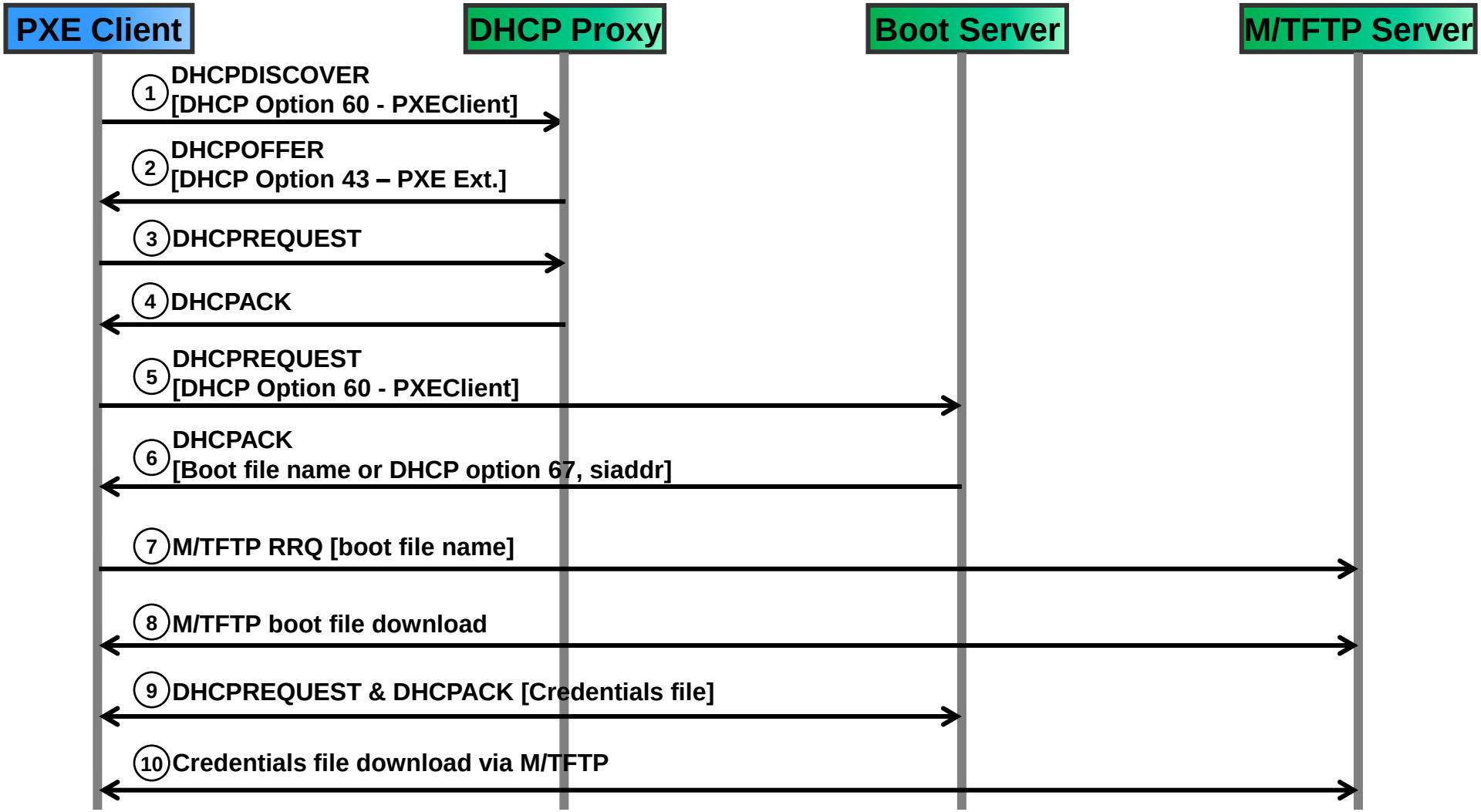
PXE is an extension of DHCP defined by Intel back in 1999 to allow computers to boot over the network, i.e. to download to operating image from a TFTP server.
PXE uses the protocols IP, UDP, TFTP and DHCP with PXE-specific extensions, i.e. PXE related information is piggy-backed onto DHCP packets.

PXE architecture:
PXE runs on the BIOS of the NIC (Network Interface Card).
PXE provides APIs to a bootstrap program for accessing the protocols involved in PXE operation.



6. PXE – Preboot Execution Environment (2/4)

PXE message flow:



6. PXE – Preboot Execution Environment (3/4)

1. DHCPDISCOVER:

Upon boot, the PXE-enabled client sends a DHCPDISCOVER packet to the UDP server port 67. This packet carries the DHCP option 60 that contains the client architecture and UNDI version (Universal Network Driver Interface).

2. & 3. & 4. DHCPOFFER (DHCPREQUEST, DHCPACK):

The PXE-enabled DHCP server (DHCP proxy) responds with a DHCPOFFER sent to the client UDP port 68.
Optionally, the client may obtain an IP address from the DHCP proxy. In that case the client must complete the DHCP transaction to be compliant with the DHCP protocol (steps 3. & 4.).
The DHCPOFFER packet contains the DHCP option 43 (vendor specific information) carrying a list of boot server IP addresses.

5. DHCPREQUEST to selected boot server:

The client sends a DHCPREQUEST with the same information as in step 1 to the selected boot server, either on broadcast UDP port 67 or multicast/unicast UDP port 4011.

6. DHCPACK with boot file name and siaddr:

The boot server responds with a DHCPACK packet containing the boot file name (file to be downloaded and booted), the IP address of a TFTP server encoded in the DHCP `siaddr` field and an optional MTFTP port number. The boot file name is either contained in the DHCP boot file name field or in DHCP option 67 (boot file option).

6. PXE – Preboot Execution Environment (4/4)

7. & 8. Boot file download through TFTP or MTFTP (Multicast TFTP):

The client downloads the boot file through TFTP (UDP port 69) or Multicast TFTP on the multicast UDP port number.

9. DHCPREQUEST for credentials file:

The client determines if boot file authentication is required. TFTP per se does not provide authentication of the downloaded file. If authentication is required, the client requests a credentials file from the previously contacted boot server. The credentials file contains some kind of checksum.
The boot server responds with a DHCPACK containing the path to the requested credentials file on the M/TFTP server.

10. Download and verification of credentials:

Finally, the PXE client downloads and verifies the credentials file.
If the verification succeeds, the client boots the downloaded boot file.