NWE
Automatic IP-Configuration
BOOTP/DHCP

v 1.ϵ

Rolf Schmutz, rschmutz@acm.org

17th November 2004

Abstract

Automatic IP-configuration not only enables diskless hosts or mobile systems to fetch the correct setup but may also help in administering large networks. BOOTP and DHCP are modern alternatives to the clumsy RARP/ICMP-method

Typeset using LaTeXand FoilTeX and, yes I know there is too much on one slide™

Fachhochschule Solothurn Nordwestschweiz
Technik-Wirtschaft-Soziales

Automatic IP-Configuration

why automatic IP-configuration?

- diskless[1] systems boot-up
- "zero configuration" for mobile systems (plug-and-play on L3-level)
- managing large installations
- prevent IP-address clashes due to manual configuration errors
- due to the central administration, changes in network-parameters (gateways, DNS-servers, etc) needs to be done only once
- better control of resources, security to some degree

[1] devices with no local storage — originally mostly X11-terminals

Before BOOTP/DHCP

- multiple, different services to retrieve all IP-configuration parameters:
  - RARP: Reverse-ARP, requires one server for each L3-network, delivers only IP-address
  - ICMP-Router-Solicitation/ICMP-Router-Advertisment: gateway(s)
  - ICMP-Netmask-Request/-Reply: Netmask for local network

downside of this method:

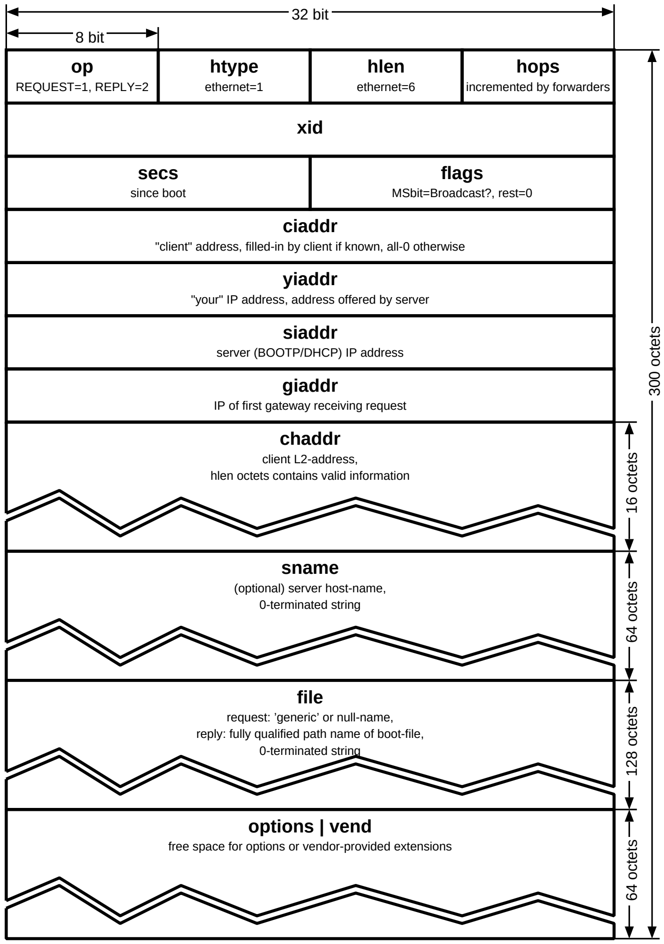
- "... retrieve *all* IP-configuration parameters"?
  What about DNS-server(s), DNS-domain-name, DNS-hostname, NTP-server, etc?
- several servers/routers must be set-up carefully to work together. Large installations with 100's of L3-networks would require unreasonable numbers of RARP-servers

BOOTP/DHCP

- BOOTP is the BOOT Protocol, RFC951 (1985) and clarifications/extensions RFC1542 (1993)
- DHCP is the Dynamic Host Configuration Protocol, RFC1531/1533/1541 (1993) and RFC2131/2132 (1997)
- Both protocols supply *all*[2] IP-configuration parameters
- Both protocols are encapsulated in UDP[3], server is listening on UDP 67 and client-requests originate from UDP 68
- ... and therefore may be forwarded by routers
- ... and therefore only a single server[4] is sufficient even for large networks
- ... this enables real central administration of IP-configuration parameters
- BOOTP was designed with extensibility[5] in mind...
- this design has proved successful, DHCP is a simple extension of the BOOTP *protocol*

[2] *all required*, that is...
[3] BOOTP/DHCP-servers may be implemented as user-space processes since socket-access is sufficient
[4] usually in a failover-configuration. Compare this to the situation using RARP
[5] ie, the packet-format is extensible such that new key/value-pairs may be defined

BOOTP: Packet Format



op operation field,request=1 and reply=2

hytpe "hardware" on L2, ethernet=1

hlen length of L2-address in octets, ethernet=6

hops number of routers, each router increments this field by one

xid transaction id: set by client, verbatim copyied by server

secs client fills in seconds since boot

ciaddr client IP-address, filled-in by client if already known

yiaddr "your IP-address", IP assigned to client by server

siaddr "server IP-address", IP of server sending the reply

giaddr "gateway IP-address", IP of receiving interface of *first* gateway forwarding the request

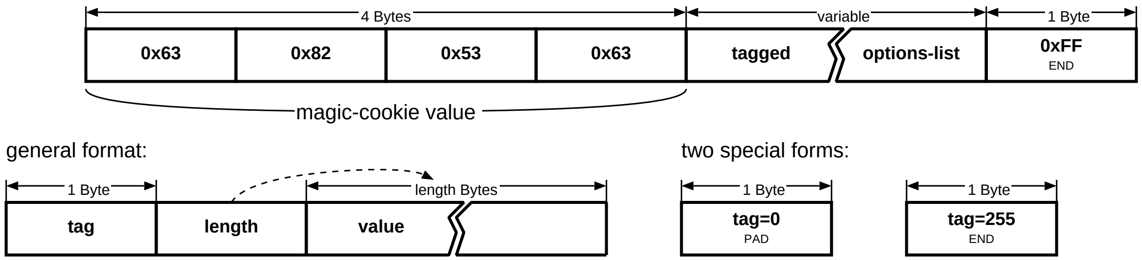
chaddr "client hardware-address", L2-address of client

sname optional "server name", filled in by server on reply

file boot file name, client may request a certain file (or "generic" if left empty) — server provides full pathname to boot-file

options|vend data-area for extensions, see slide 5

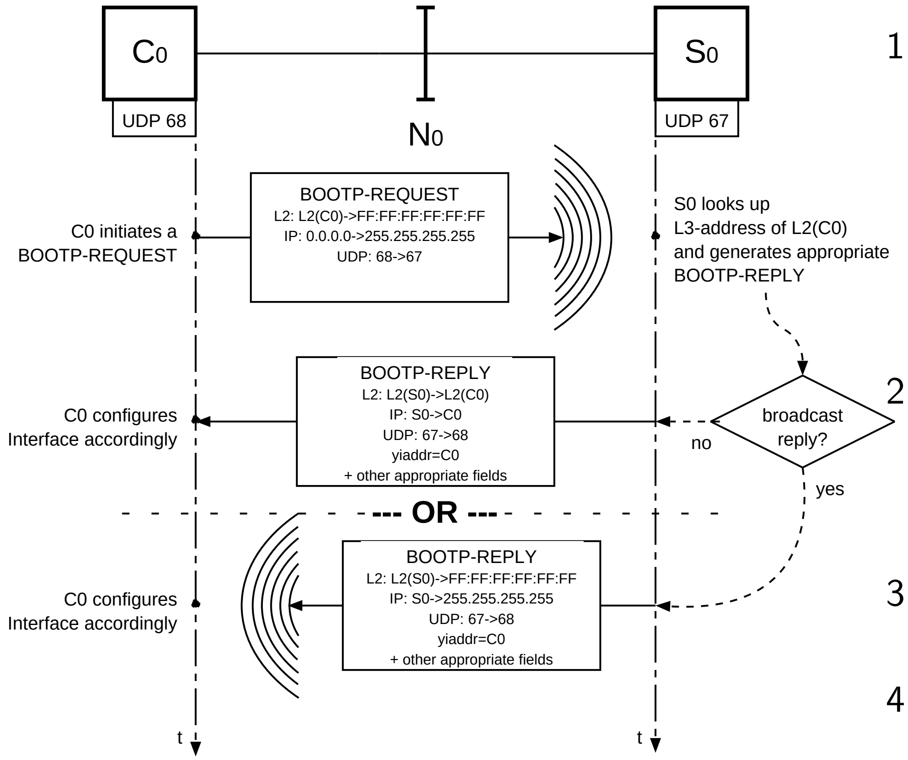
BOOTP: options|vend-Field



Some options:

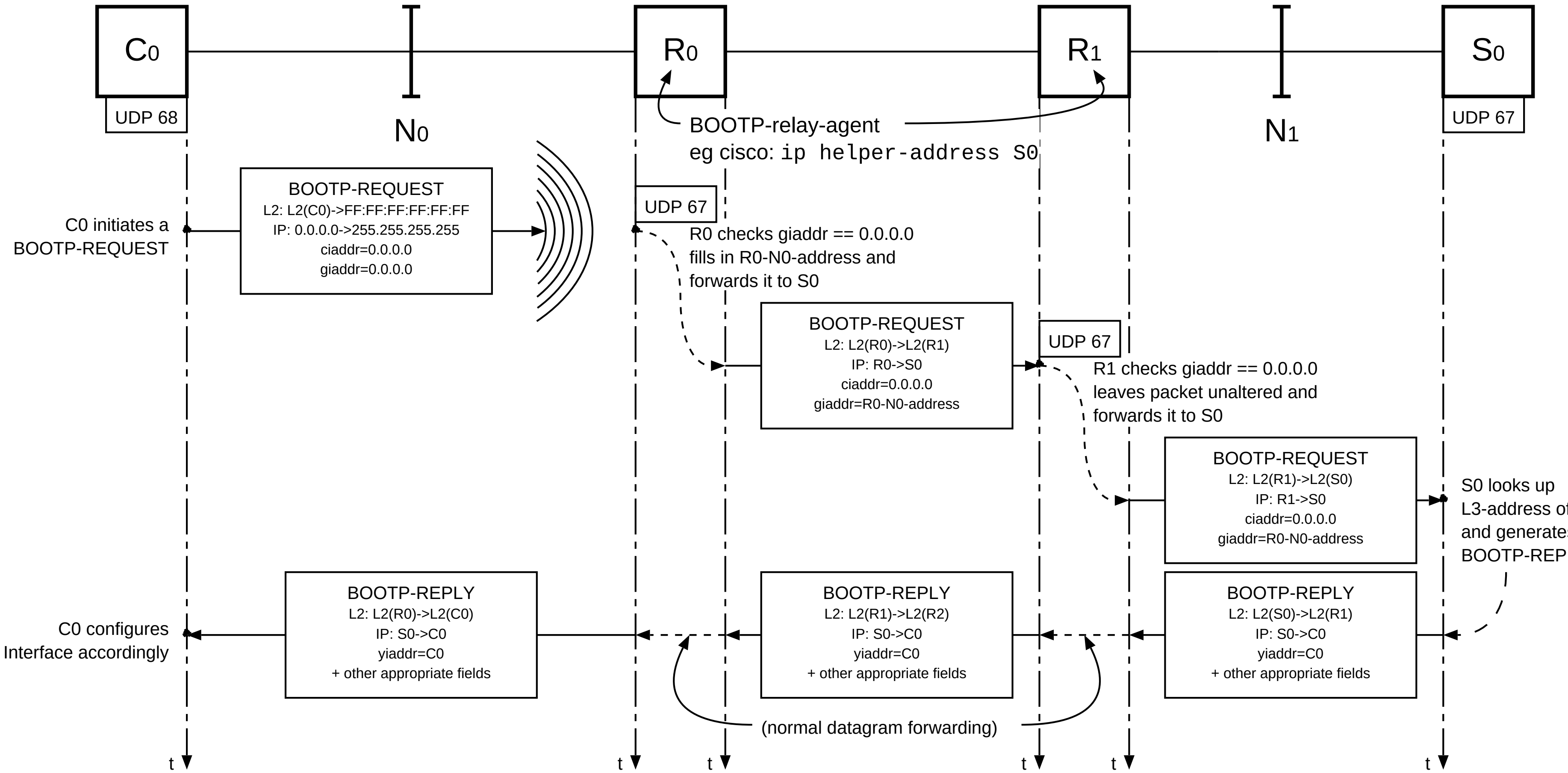
Description	Tag	Length
Padding	0	-
Subnet Mask	1	4
Time Offset	2	4
Default Routers	3	variable
Time Servers	4	variable
DNS Servers	6	variable
Print Servers	9	variable
Host Name	12	variable
Vendor Specific	128...254	variable
End Of Options	255	-

BOOTP: Communication



1. the client initiates a BOOTP-REQUEST
   op=1, htype/hlen/chaddr=actual values,
   xid="unique-value' ' , secs=0,
   flags[15]=1 if reply should be broadcasted
   all other fields=0 or RFC-initial-values
   src port=68, dst port=67,
   the BOOTP-REQUEST is broadcast (L2/L3)
2. server listens on UDP-67,
   on reception of a request, the L3-address
   of the client is looked up using L2-address
   from chaddr as a key
3. if the flag[15]-bit is set, the reply is broadcasted
   otherwise the reply is sent unicast L2/L3
4. the client may now configure the interface appropriately

BOOTP: Relay-Communication



BOOTP: Implementations

"pure" BOOTP-servers are rare nowadays, some UNIX systems are shipped with `bootpd`[6]. Unfortunately the configuration-database used by `bootpd` is uncanny ugly[7]:

```
# Sample bootptab file (domain=andrew.cmu.edu)
.default:\
     :hd=/usr/boot:bf=null:\
     :ds=netserver, lancaster:\
     :sa=pcs2, pcs1:\
     :ts=pcs2, pcs1:\
     :sm=255.255.255.0:\
     :gw=gw.cs.cmu.edu:\
     :hn:to=-18000:

carnegie:ht=6:ha=7FF8100000AF:tc=.default:
baldwin:ht=1:ha=0800200159C3:tc=.default:
arnold:ht=1:ha=0800200102AD:tc=.default:
...
```

[6] see `man bootpd`, a better implementation of BOOTP/DHCP is described in the DHCP-section

[7] `man 5 bootptab`, resembles the infamous `printcap` or `termcap` file format

BOOTP: TFTP

BOOTP may be used as a first stage in a multi-stage boot process. After the initial BOOTP-REQUEST/REPLY, most of the IP-configuration is determined.
Diskless systems however, may need a system-image[8] or additional configuration data[9] to continue the boot process.
The `file`-field together with the `sa`-tag[10] provides a flexible solution for this problem:

- the BOOTP-server fills-in the `file`-field (eg `linux.boot`) and sets the `sa`-option to the IP-address of a TFTP-server[11].
- after the client receives the BOOTP-reply, it configures its interfaces and connects to the TFTP-server to fetch the appropriate file
- the file may contain program code (execution of this code may start the kernel, etc) or additional configuration data

---

8 kernel or next-stage bootloader
9 cisco IOS-configuration file
10 tftp-server, see `man 5 bootptab` or IANA.org
11 Trivial File Transfer Protocol, a simple FTP based on UDP

DHCP

BOOTP is wonderful and works great, but there are situations where a more sophisticated solution would be appropriate:

- mobile systems: notebook computers with the need for temporal network access. Some machines will only show up one-in-a-life-time, others regurarly[12]
- limited IP-address space: for some reason there are more hosts than IP-addresses.[13] Dynamic management of the address space handles such situations gracefully[14]
- "zero configuration" network — no maintenance of the BOOTP-database (file)

[12] using BOOTP in such a case would allocate an IP regardless of the frequency of usage — and keep it allocated until the end of days
[13] this should not happen when using RFC1918 addresses
[14] eg dial-up ISPs

DHCP: Overview

DHCP provides these additional features:

- dynamic allocation: IP-addresses may be dynamically allocated (a *lease* in DHCP-parlance)
- entire ranges of IP-addresses may be declared as a dynamic pool
- operation mode includes:
  - static (like BOOTP)
  - automatic like BOOTP, but without configuration, IP remains allocated until the end of days)
  - dynamic (DHCP, temporal limited IP allocation)

  these modes may be used simultaneous
- DHCP is stateful on both server- and client-side, see slide 16 (BOOTP, in contrast, is stateless on the server side)
- dynamic allocated IPs are valid only for the duration of the *lease-time*. The *lease* on the IP must be renewed by the client before it times out
- therefore IPs may be leased to different hosts at different times
- DHCP is interoperable, BOOTP-clients are served as well. . .

DHCP: Packet Format

like BOOTP[15]

[15] :) . . . only the options-section has grown a little

DHCP: Special Tags

DHCP defines some special-purpose tags for use in the data-section of the PDU[16]. In fact, the presence of these special tags distinguishes DHCP from BOOTP:

Tag	Description	Length	Values
T51	Lease Time	4	seconds
T53	Message Type	1	1=DISCOVER, 2=OFFER, . . .
T54	Server Identifier	4	servers IP-address
T55	Parameter Request List	var	list of options (tags) requested
T58	Renewal Timespan (T1) (lease time)	4	time in seconds
T59	Rebind Timespan (T2)	4	time in seconds

See slide 16 for T53 operation

[16] see RFC2132

DHCP: Packet Dump 1/2

"uninitialized source IP" "limited broadcast"

```
23:53:00.175176 0.0.0.0.bootpc > 255.255.255.255.bootps:  xid:0x2999cf79
        vend-rfc1048 DHCP:DISCOVER PR:SM+DG+NS
0x0000   4500 0148 f9a8 0000 ff11 c0fc 0000 0000        E..H............
0x0010   ffff ffff 0044 0043 0134 65fd 0101 0600        .....D.C.4e.....
0x0020   2999 cf79 0000 0000 0000 0000 0000 0000        )..y............
0x0030   0000 0000 0000 0000 0030 6500 ecff 0000        .........0e.....
0x0040   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0050   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0070   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0080   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0090   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00a0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00b0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00c0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00d0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00e0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00f0   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0100   0000 0000 0000 0000 6382 5363 3501 0137        ........c.Sc5..7
0x0110   0901 0306 0f70 714e 4f5f 3902 05dc 3d06        .....pqNO_9...=.
0x0120   0073 6c69 636b 3304 0076 a700 0c05 736c        .slick3..v....sl
0x0130   6963 6bff 0000 0000 0000 0000 0000 0000        ick.............
0x0140   0000 0000 0000 0000                            ........
```

unused BOOTP-fields

magic cookie

T53=DHCP Operation

T55=Parameter Req List

...

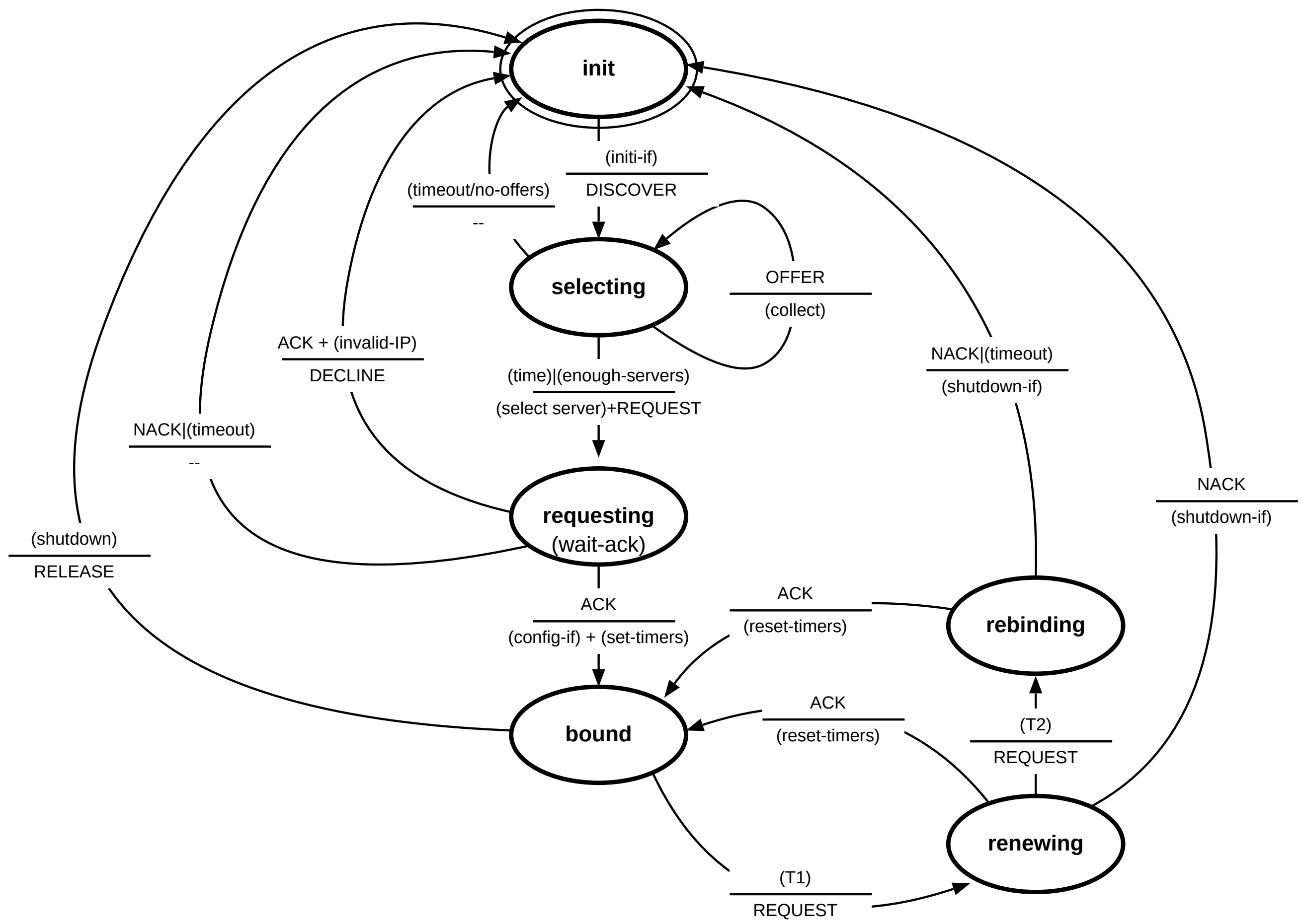
etc, etc...

DHCP: Packet Dump 2/2

- the source IP `0.0.0.0` is used if the *sender* IP-address is not (yet) known
- the *limited broadcast* address is used as destination
- tagged-options are all listed in the *options*|*vend*-field
- wasted space (empty server- and file-name fields) may also be used for tag-option storage[17]

[17] find the appropriate tag in RFC2131

DHCP: State-Transition Diagram 1/3



DHCP: State Transition Diagram 2/3

. . . there is one additional operation (ie. *action*) not present in the diagram:

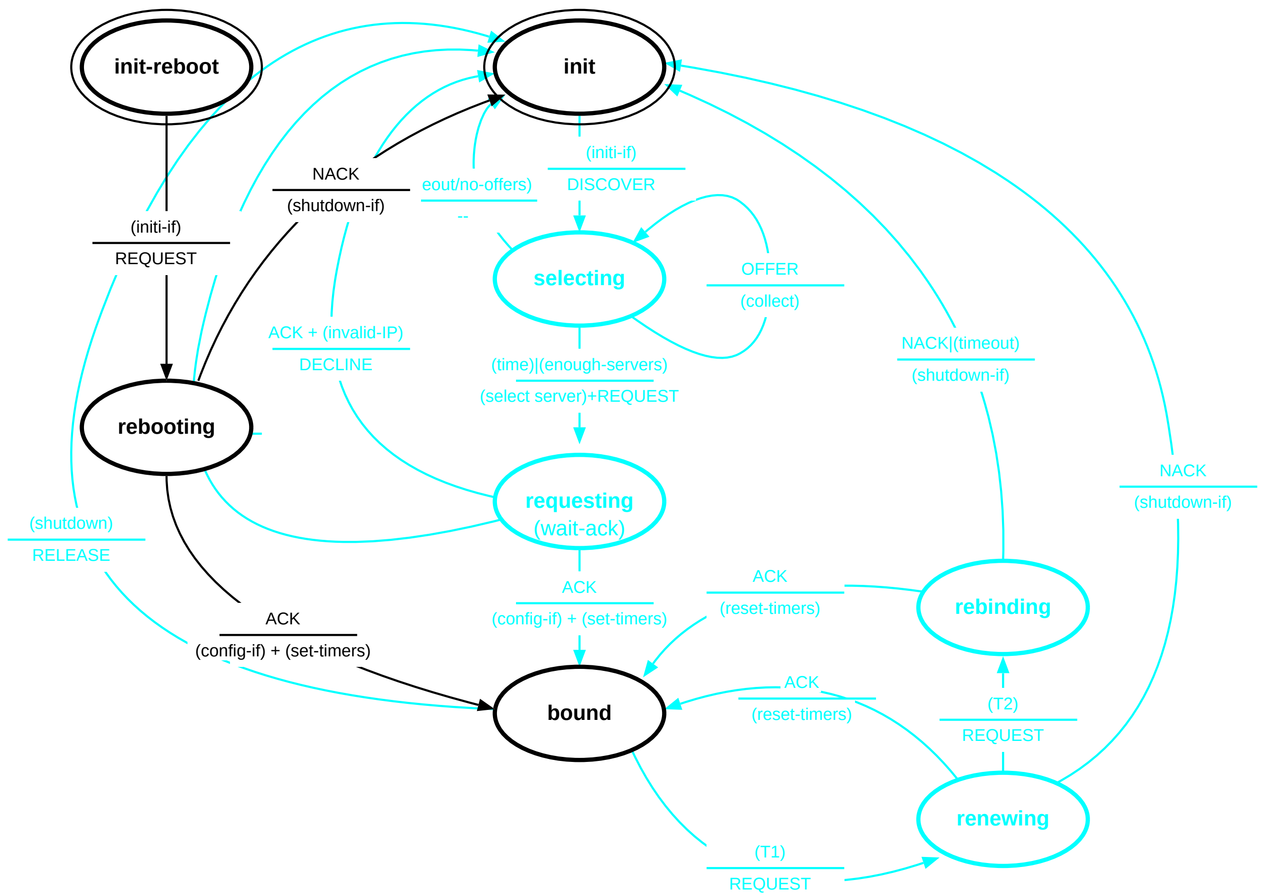
- `DHCPINFORM` may be used to just get some information from the server, *without* the server keeping-state of the client

this may be used to retrieve additional options (T55) if the IP is already known

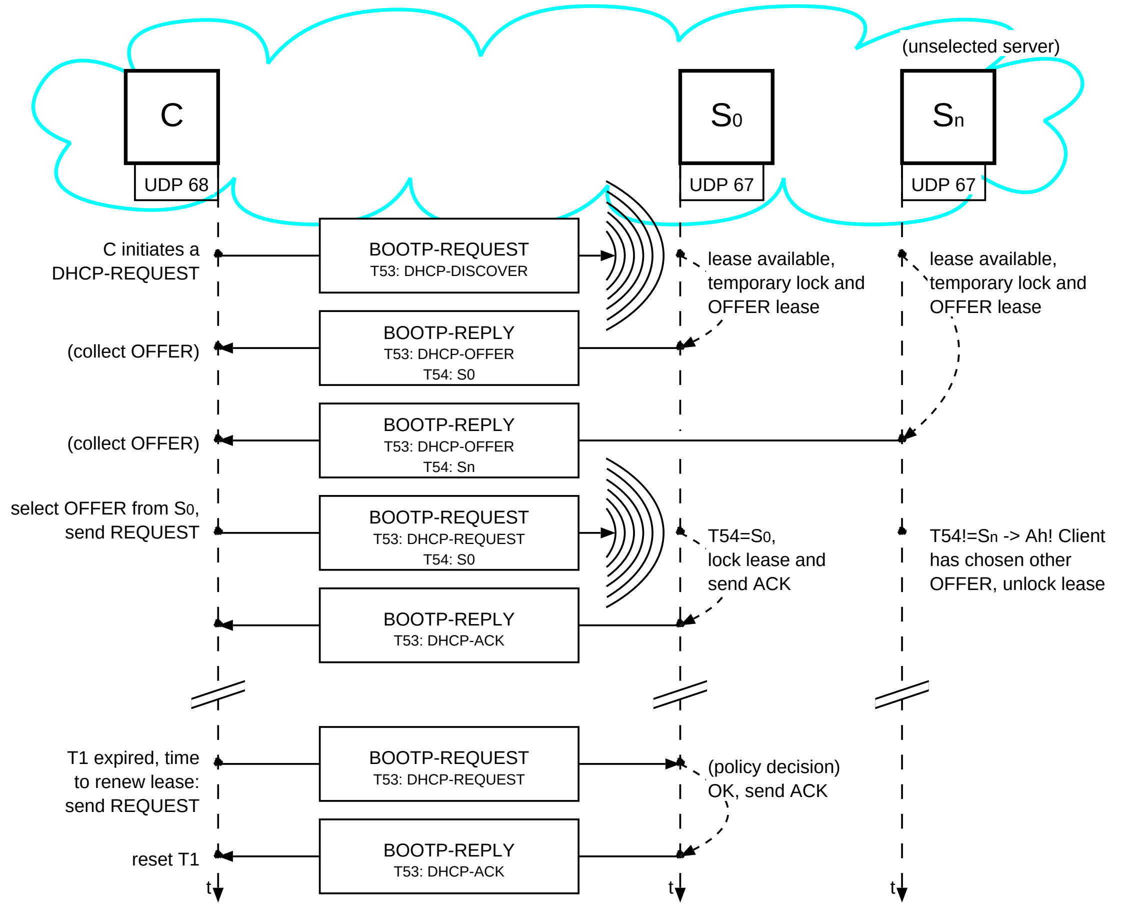
Hosts may operate stateful over reboots[18] and choose the "fast-route" on reboot:

[18] ie, storing the lease-information on disk

DHCP: State Transition Diagram 3/3



DHCP: Operation 1/2



DHCP: Implementations

Most OS comes with a built-in DHCP-client-implementation.[19] DHCP requires no special APIs for proper operation and may therefore be implemented in user space.[20]

The reference implementation for both client- and server-DHCP can be found on `http://www.isc.org/` — this is a very capable server implementation, by the way. . .

Linux systems may come with `dhclient`, `dhcp-client`, `pump`, etc. Choose the one that comes with your distro[21])

[19] some may even let you choose between BOOTP or DHCP

[20] `root` required for socket access

[21] the ISC-client lets you inspect the lease under `/var/run/dhcpd/leases`