Clarifications and Extensions for the Bootstrap Protocol

Status of this Memo

   This RFC specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" for the standardization state and status
   of this protocol. Distribution of this memo is unlimited.

Abstract

   Some aspects of the BOOTP protocol were rather loosely defined in its
   original specification. In particular, only a general description
   was provided for the behavior of "BOOTP relay agents" (originally
   called BOOTP forwarding agents"). The client behavior description
   also suffered in certain ways. This memo attempts to clarify and
   strengthen the specification in these areas. Due to some errors
   introduced into RFC 1532 in the editorial process, this memo is
   reissued as RFC 1542.

   In addition, new issues have arisen since the original specification
   was written. This memo also attempts to address some of these.

Table of Contents

1. Introduction

   The Bootstrap Protocol (BOOTP) is a UDP/IP-based protocol which
   allows a booting host to configure itself dynamically and without
   user supervision. BOOTP provides a means to notify a host of its
   assigned IP address, the IP address of a boot server host, and the
   name of a file to be loaded into memory and executed [1]. Other
   configuration information such as the local subnet mask, the local
   time offset, the addresses of default routers, and the addresses of
   various Internet servers can also be communicated to a host using
   BOOTP [2].

   Unfortunately, the original BOOTP specification [1] left some issues
   of the protocol open to question. The exact behavior of BOOTP relay
   agents formerly called "BOOTP forwarding agents") was not clearly
   specified. Some parts of the overall protocol specification actually
   conflict, while other parts have been subject to misinterpretation,
   indicating that clarification is needed. This memo addresses these
   problems.

   Since the introduction of BOOTP, the IEEE 802.5 Token Ring Network
   has been developed which presents a unique problem for BOOTP's
   particular message-transfer paradigm. This memo also suggests a
   solution for this problem.

   NOTE: Unless otherwise specified in this document or a later
   document, the information and requirements specified througout this
   document also apply to extensions to BOOTP such as the Dynamic Host
   Configuration Protocol (DHCP) [3].

1.1 Requirements

   In this memo, the words that are used to define the significance of
   particular requirements are capitalized. These words are:

      o "MUST"

        This word or the adjective "REQUIRED" means that the item
        is an absolute requirement of the specification.

      o "MUST NOT"

        This phrase means that the item is an absolute prohibition
        of the specification.

      o "SHOULD"

        This word or the adjective "RECOMMENDED" means that there
        may exist valid reasons in particular circumstances to
        ignore this item, but the full implications should be
        understood and the case carefully weighed before choosing a
        different course.

      o "SHOULD NOT"

        This phrase means that there may exist valid reasons in
        particular circumstances when the listed behavior is
        acceptable or even useful, but the full implications should
        be understood and the case carefully weighed before
        implementing any behavior described with this label.

      o "MAY"

        This word or the adjective "OPTIONAL" means that this item
        is truly optional. One vendor may choose to include the
        item because a particular marketplace requires it or
        because it enhances the product, for example; another
        vendor may omit the same item.

1.2 Terminology

   This memo uses the following terms:

      BOOTREQUEST

        A BOOTREQUEST message is a BOOTP message sent from a BOOTP
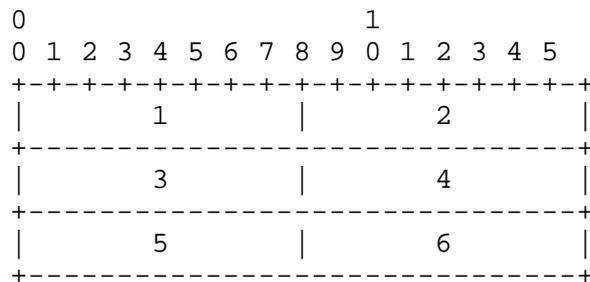   client to a BOOTP server, requesting configuration information.

BOOTREPLY

A BOOTREPLY message is a BOOTP message sent from a BOOTP server
to a BOOTP client, providing configuration information.

Silently discard

This memo specifies several cases where a BOOTP entity is to
"silently discard" a received BOOTP message. This means that
the entity is to discard the message without further
processing, and that the entity will not send any ICMP error
message as a result. However, for diagnosis of problems, the
entity SHOULD provide the capability of logging the error,
including the contents of the silently-discarded message, and
SHOULD record the event in a statistics counter.

1.3 Data Transmission Order

The order of transmission of the header and data described in this
document is resolved to the octet level. Whenever a diagram shows a
group of octets, the order of transmission of those octets is the
normal order in which they are read in English. For example, in the
following diagram, the octets are transmitted in the order they are
numbered.

```
              0                   1
              0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
             |       1       |       2       |
             +-----------------------------+
             |       3       |       4       |
             +-----------------------------+
             |       5       |       6       |
             +-----------------------------+
```

Whenever an octet represents a numeric quantity, the leftmost bit in
the diagram is the high order or most significant bit. That is, the
bit labeled 0 is the most significant bit. For example, the
following diagram represents the value 170 (decimal).

```
                0 1 2 3 4 5 6 7
               +-+-+-+-+-+-+-+-+
               |1 0 1 0 1 0 1 0|
               +--------------+
```
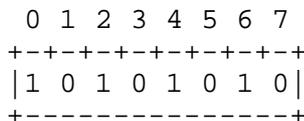
Similarly, whenever a multi-octet field represents a numeric quantity
the leftmost bit of the whole field is the most significant bit.
When a multi-octet quantity is transmitted the most significant octet

is transmitted first.

2. General Issues

   This section covers issues of general relevance to all BOOTP entities
   (clients, servers, and relay agents).

2.1 General BOOTP Processing

   The following consistency checks SHOULD be performed on BOOTP
   messages:

      o The IP Total Length and UDP Length must be large enough to
        contain the minimal BOOTP header of 300 octets (in the UDP
        data field) specified in [1].

   NOTE: Future extensions to the BOOTP protocol may increase the size
   of BOOTP messages. Therefore, BOOTP messages which, according to the
   IP Total Length and UDP Length fields, are larger than the minimum
   size specified by [1] MUST also be accepted.

      o The 'op' (opcode) field of the message must contain either the
        code for a BOOTREQUEST (1) or the code for a BOOTREPLY (2).

   BOOTP messages not meeting these consistency checks MUST be silently
   discarded.

2.2 Definition of the 'flags' Field

   The standard BOOTP message format defined in [1] includes a two-octet
   field located between the 'secs' field and the 'ciaddr' field. This
   field is merely designated as "unused" and its contents left
   unspecified, although Section 7.1 of [1] does offer the following
   suggestion:

      "Before setting up the packet for the first time, it is a good
      idea to clear the entire packet buffer to all zeros; this will
      place all fields in their default state."

      This memo hereby designates this two-octet field as the 'flags'
      field.

      This memo hereby defines the most significant bit of the 'flags'
      field as the BROADCAST (B) flag. The semantics of this flag are
      discussed in Sections 3.1.1 and 4.1.2 of this memo.

      The remaining bits of the 'flags' field are reserved for future
      use. They MUST be set to zero by clients and ignored by servers

and relay agents.

The 'flags' field, then, appears as follows:

```
            0                   1
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |B|              MBZ            |
            +-+----------------------------+
```
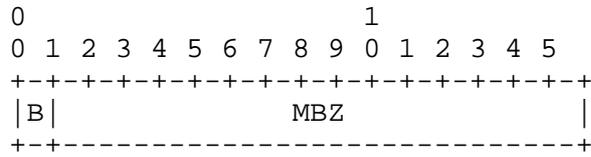
where:

   B   BROADCAST flag (discussed in Sections 3.1.1 and 4.1.2)

   MBZ   MUST BE ZERO (reserved for future use)

The format of a BOOTP message is shown below. The numbers in
parentheses indicate the size of each field in octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     op (1)    |    htype (1)  |    hlen (1)   |    hops (1)   |
+---------------+---------------+---------------+---------------+
|                            xid (4)                           |
+-------------------------------+-------------------------------+
|           secs (2)            |            flags (2)          |
+-------------------------------+-------------------------------+
|                          ciaddr  (4)                         |
+--------------------------------------------------------------+
|                          yiaddr  (4)                         |
+--------------------------------------------------------------+
|                          siaddr  (4)                         |
+--------------------------------------------------------------+
|                          giaddr  (4)                         |
+--------------------------------------------------------------+
|                                                              |
|                          chaddr  (16)                        |
|                                                              |
|                                                              |
+--------------------------------------------------------------+
|                                                              |
|                          sname   (64)                        |
+--------------------------------------------------------------+
|                                                              |
|                          file    (128)                       |
+--------------------------------------------------------------+
|                                                              |
|                          vend    (64)                        |
+--------------------------------------------------------------+
```
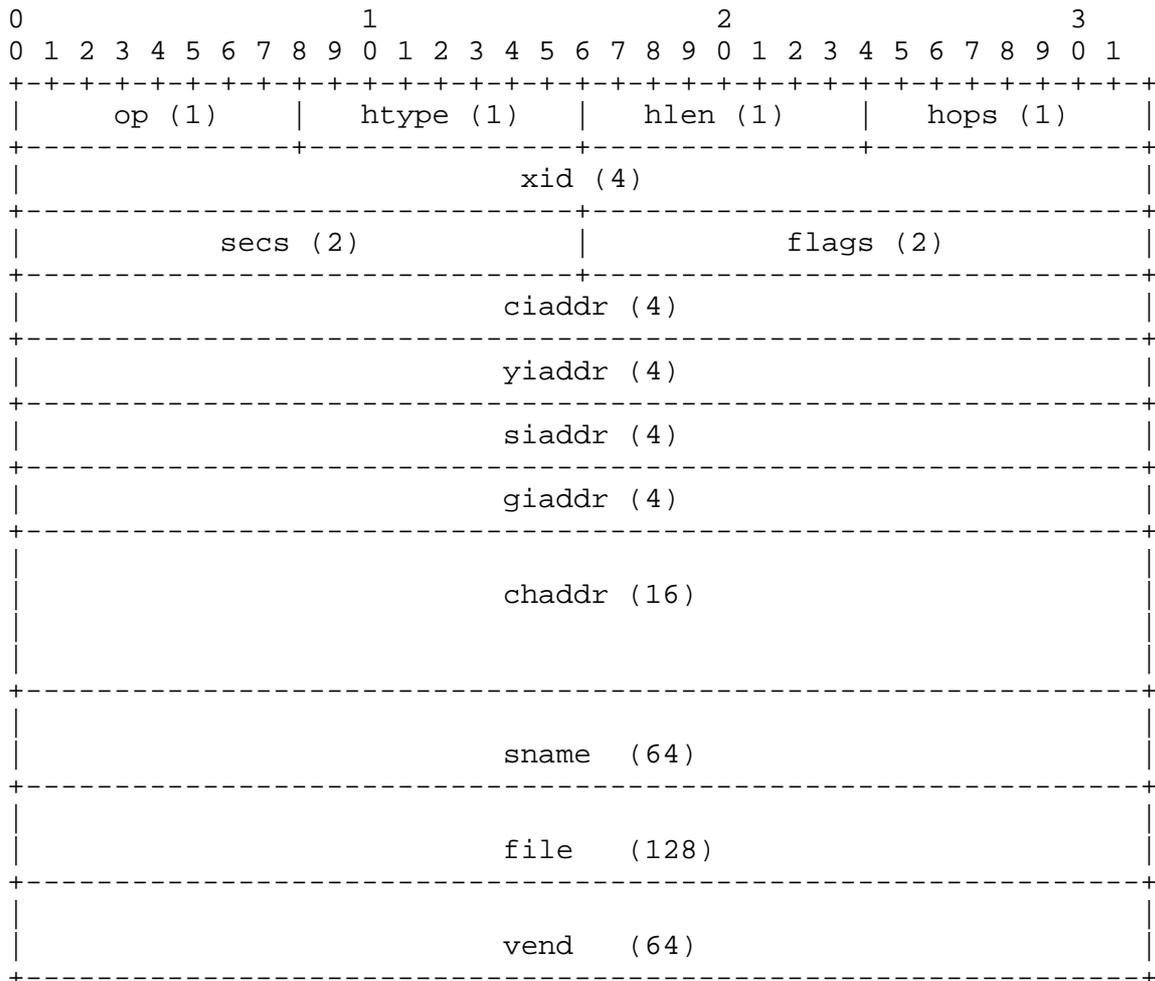
2.3 Bit Ordering of Hardware Addresses

   The bit ordering used for link-level hardware addresses in the
   'chaddr' field SHOULD be the same as the ordering used for the ARP
   protocol [4] on the client's link-level network (assuming ARP is
   defined for that network).

   The 'chaddr' field MUST be preserved as it was specified by the BOOTP
   client. A relay agent MUST NOT reverse the bit ordering of the
   'chaddr' field even if it happens to be relaying a BOOTREQUEST
   between two networks which use different bit orderings.

      DISCUSSION:

         One of the primary reasons the 'chaddr' field exists is to
         enable BOOTP servers and relay agents to communicate directly

with clients without the use of broadcasts. In practice, the
contents of the 'chaddr' field is often used to create an ARP-
cache entry in exactly the same way the normal ARP protocol
would have. Clearly, interoperability can only be achieved if
a consistent interpretation of the 'chaddr' field is used.

As a practical example, this means that the bit ordering used
for the 'chaddr' field by a BOOTP client on an IEEE 802.5 Token
Ring network is the opposite of the bit ordering used by a
BOOTP client on a DIX ethernet network.

2.4 BOOTP Over IEEE 802.5 Token Ring Networks

   Special consideration of the client/server and client/relay agent
   interactions must be given to IEEE 802.5 networks because of non-
   transparent bridging.

   The client SHOULD send its broadcast BOOTREQUEST with an All Routes
   Explorer RIF. This will enable servers/relay agents to cache the
   return route if they choose to do so. For those server/relay agents
   which cannot cache the return route (because they are stateless, for
   example), the BOOTREPLY message SHOULD be sent to the client's
   hardware address, as taken from the BOOTP message, with a Spanning
   Tree Rooted RIF. The actual bridge route will be recorded by the
   client and server/relay agent by normal ARP processing code.

      DISCUSSION:

         In the simplest case, an unbridged, single ring network, the
         broadcast behavior of the BOOTP protocol is identical to that
         of Ethernet networks. However, a BOOTP client cannot know, a
         priori, that an 802.5 network is not bridged. In fact, the
         likelihood is that the server, or relay agent, will not know
         either.

         Of the four possible scenerios, only two are interesting: where
         the assumption is that the 802.5 network is not bridged and it
         is, and the assumption that the network is bridged and it is
         not. In the former case, the Routing Information Field (RIF)
         will not be used; therefore, if the server/relay agent are on
         another segment of the ring, the client cannot reach it. In
         the latter case, the RIF field will be used, resulting in a few
         extraneous bytes on the ring. It is obvious that an almost
         immeasurable inefficiency is to be preferred over a complete
         failure to communicate.

Given that the assumption is that RIF fields will be needed, it is necesary to determine the optimum method for the client to reach the server/relay agent, and the optimum method for the response to be returned.

3. BOOTP Client Behavior

This section clarifies various issues regarding BOOTP client behavior.

3.1 Client use of the 'flags' field

3.1.1 The BROADCAST flag

Normally, BOOTP servers and relay agents attempt to deliver BOOTREPLY messages directly to a client using unicast delivery. The IP destination address (in the IP header) is set to the BOOTP 'yiaddr' address and the link-layer destination address is set to the BOOTP 'chaddr' address. Unfortunately, some client implementations are unable to receive such unicast IP datagrams until they know their own IP address (thus we have a "chicken and egg" issue). Often, however, they can receive broadcast IP datagrams (those with a valid IP broadcast address as the IP destination and the link-layer broadcast address as the link-layer destination).

If a client falls into this category, it SHOULD set (to 1) the newly-defined BROADCAST flag in the 'flags' field of BOOTREPLY messages it generates. This will provide a hint to BOOTP servers and relay agents that they should attempt to broadcast their BOOTREPLY messages to the client.

If a client does not have this limitation (i.e., it is perfectly able to receive unicast BOOTREPLY messages), it SHOULD NOT set the BROADCAST flag (i.e., it SHOULD clear the BROADCAST flag to 0).

DISCUSSION:

This addition to the protocol is a workaround for old host implementations. Such implementations SHOULD be modified so that they may receive unicast BOOTREPLY messages, thus making use of this workaround unnecessary. In general, the use of this mechanism is discouraged.

3.1.2 The remainder of the 'flags' field

The remaining bits of the 'flags' field are reserved for future use. A client MUST set these bits to zero in all BOOTREQUEST messages it generates. A client MUST ignore these bits in all BOOTREPLY messages

it receives.

3.2 Definition of the 'secs' field

   The 'secs' field of a BOOTREQUEST message SHOULD represent the
   elapsed time, in seconds, since the client sent its first BOOTREQUEST
   message. Note that this implies that the 'secs' field of the first
   BOOTREQUEST message SHOULD be set to zero.

   Clients SHOULD NOT set the 'secs' field to a value which is constant
   for all BOOTREQUEST messages.

      DISCUSSION:

         The original definition of the 'secs' field was vague. It was
         not clear whether it represented the time since the first
         BOOTREQUEST message was sent or some other time period such as
         the time since the client machine was powered-up. This has
         limited its usefulness as a policy control mechanism for BOOTP
         servers and relay agents. Furthermore, certain client
         implementations have been known to simply set this field to a
         constant value or use incorrect byte-ordering. Incorrect
         byte-ordering usually makes it appear as if a client has been
         waiting much longer than it really has, so a relay agent will
         relay the BOOTREQUEST sooner than desired (usually
         immediately). These implementation errors have further
         undermined the usefulness of the 'secs' field. These incorrect
         implementations SHOULD be corrected.

3.3 Use of the 'ciaddr' and 'yiaddr' fields

   If a BOOTP client does not know what IP address it should be using,
   the client SHOULD set the 'ciaddr' field to 0.0.0.0. If the client
   has the ability to remember the last IP address it was assigned, or
   it has been preconfigured with an IP address via some alternate
   mechanism, the client MAY fill the 'ciaddr' field with that IP
   address. If the client does place a non-zero IP address in the
   'ciaddr' field, the client MUST be prepared to accept incoming
   unicast datagrams addressed to that IP address and also answer ARP
   requests for that IP address (if ARP is used on that network).

   The BOOTP server is free to assign a different IP address (in the
   'yiaddr' field) than the client expressed in 'ciaddr'. The client
   SHOULD adopt the IP address specified in 'yiaddr' and begin using it
   as soon as possible.

DISCUSSION:

There are various interpretations about the purpose of the
'ciaddr' field and, unfortunately, no agreement on a single
correct interpretation. One interpretation is that if a client
is willing to accept whatever IP address the BOOTP server
assigns to it, the client should always place 0.0.0.0 in the
'ciaddr' field, regardless of whether it knows its previously-
assigned address. Conversely, if the client wishes to assert
that it must have a particular IP address (e.g., the IP address
was hand-configured by the host administrator and BOOTP is only
being used to obtain a boot file and/or information from the
'vend' field), the client will then fill the 'ciaddr' field
with the desired IP address and ignore the IP address assigned
by the BOOTP server as indicated in the 'yiaddr' field. An
alternate interpretation holds that the client always fills the
'ciaddr' field with its most recently-assigned IP address (if
known) even if that address may be incorrect. Such a client
will still accept and use the address assigned by the BOOTP
server as indicated in the 'yiaddr' field. The motivation for
this interpretation is to aid the server in identifying the
client and/or in delivering the BOOTREPLY to the client. Yet a
third (mis)interpretation allows the client to use 'ciaddr' to
express the client's desired IP address, even if the client has
never used that address before or is not currently using that
address.

The last interpretation is incorrect as it may prevent the
BOOTREPLY from reaching the client. The server will usually
unicast the reply to the address given in 'ciaddr' but the
client may not be listening on that address yet, or the client
may be connected to an incorrect subnet such that normal IP
routing (correctly) routes the reply to a different subnet.

The second interpretation also suffers from the "incorrect
subnet" problem.

The first interpretation seems to be the safest and most likely
to promote interoperability.

3.4 Interpretation of the 'giaddr' field

The 'giaddr' field is rather poorly named. It exists to facilitate
the transfer of BOOTREQUEST messages from a client, through BOOTP
relay agents, to servers on different networks than the client.
Similarly, it facilitates the delivery of BOOTREPLY messages from the
servers, through BOOTP relay agents, back to the client. In no case
does it represent a general IP router to be used by the client. A

BOOTP client MUST set the 'giaddr' field to zero (0.0.0.0) in all
BOOTREQUEST messages it generates.

A BOOTP client MUST NOT interpret the 'giaddr' field of a BOOTREPLY
message to be the IP address of an IP router. A BOOTP client SHOULD
completely ignore the contents of the 'giaddr' field in BOOTREPLY
messages.

   DISCUSSION:

      The semantics of the 'giaddr' field were poorly defined.
      Section 7.5 of [1] states:

         "If 'giaddr' (gateway address) is nonzero, then the packets
         should be forwarded there first, in order to get to the
         server."

In that sentence, "get to" refers to communication from the client to
the server subsequent to the BOOTP exchange, such as a TFTP session.
Unfortunately, the 'giaddr' field may contain the address of a BOOTP
relay agent that is not itself an IP router (according to [1],
Section 8, fifth paragraph), in which case, it will be useless as a
first-hop for TFTP packets sent to the server (since, by definition,
non-routers don't forward datagrams at the IP layer).

Although now prohibited by Section 4.1.1 of this memo, the 'giaddr'
field might contain a broadcast address according to Section 8, sixth
paragraph of [1]. Not only would such an address be useless as a
router address, it might also cause the client to ARP for the
broadcast address (since, if the client didn't receive a subnet mask
in the BOOTREPLY message, it would be unable to recognize a subnet
broadcast address). This is clearly undesirable.

To reach a non-local server, clients can obtain a first-hop router
address from the "Gateway" subfield of the "Vendor Information
Extensions" [2] (if present), or via the ICMP router discovery
protocol [5] or other similar mechanism.

3.5 Vendor information "magic cookie"

It is RECOMMENDED that a BOOTP client always fill the first four
octets of the 'vend' (vendor information) field of a BOOTREQUEST with
a four-octet identifier called a "magic cookie." A BOOTP client
SHOULD do this even if it has no special information to communicate
to the BOOTP server using the 'vend' field. This aids the BOOTP
server in determining what vendor information format it should use in
its BOOTREPLY messages.

If a special vendor-specific magic cookie is not being used, a BOOTP
client SHOULD use the dotted decimal value 99.130.83.99 as specified
in [2]. In this case, if the client has no information to
communicate to the server, the octet immediately following the magic
cookie SHOULD be set to the "End" tag (255) and the remaining octets
of the 'vend' field SHOULD be set to zero.

   DISCUSSION:

   Sometimes different operating systems or networking packages
   are run on the same machine at different times (or even at the
   same time!). Since the hardware address placed in the 'chaddr'
   field will likely be the same, BOOTREQUESTs from completely
   different BOOTP clients on the same machine will likely be
   difficult for a BOOTP server to differentiate. If the client
   includes a magic cookie in its BOOTREQUESTs, the server will at
   least know what format the client expects and can understand in
   corresponding BOOTREPLY messages.

4. BOOTP Relay Agents

   In many cases, BOOTP clients and their associated BOOTP
   server(s) do not reside on the same IP network or subnet. In
   such cases, some kind of third-party agent is required to
   transfer BOOTP messages between clients and servers. Such an
   agent was originally referred to as a "BOOTP forwarding agent."
   However, in order to avoid confusion with the IP forwarding
   function of an IP router, the name "BOOTP relay agent" is
   hereby adopted instead.

   DISCUSSION:

   A BOOTP relay agent performs a task which is distinct from an
   IP router's normal IP forwarding function. While a router
   normally switches IP datagrams between networks more-or-less
   transparently, a BOOTP relay agent may more properly be thought
   to receive BOOTP messages as a final destination and then
   generate new BOOTP messages as a result. It is incorrect for a
   relay agent implementation to simply forward a BOOTP message
   "straight through like a regular packet."

   This relay-agent functionality is most conveniently located in
   the routers which interconnect the clients and servers, but may
   alternatively be located in a host which is directly connected
   to the client subnet.

   Any Internet host or router which provides BOOTP relay-agent
   capability MUST conform to the specifications in this memo.

4.1 General BOOTP Processing for Relay Agents

   All locally delivered UDP messages whose UDP destination port number
   is BOOTPS (67) are considered for special processing by the host or
   router's logical BOOTP relay agent.

   In the case of a host, locally delivered datagrams are simply all
   datagrams normally received by that host, i.e., broadcast and
   multicast datagrams as well as unicast datagrams addressed to IP
   addresses of that host.

   In the case of a router, locally delivered datagrams are broadcast
   and multicast datagrams as well as unicast datagrams addressed to IP
   addresses of that router. These are datagrams for which the router
   should be considered an end destination as opposed to an intermediate
   switching node. Thus a unicast datagram with an IP destination not
   matching any of the router's IP addresses is not considered for
   processing by the router's logical BOOTP relay agent.

   Hosts and routers are usually required to silently discard incoming
   datagrams containing illegal IP source addresses. This is generally
   known as "Martian address filtering." One of these illegal addresses
   is 0.0.0.0 (or actually anything on network 0). However, hosts or
   routers which support a BOOTP relay agent MUST accept for local
   delivery to the relay agent BOOTREQUEST messages whose IP source
   address is 0.0.0.0. BOOTREQUEST messages from legal IP source
   addresses MUST also be accepted.

   A relay agent MUST silently discard any received UDP messages whose
   UDP destination port number is BOOTPC (68).

      DISCUSSION:

         There should be no need for a relay agent to process messages
         addressed to the BOOTPC port. Careful reading of the original
         BOOTP specification [1] will show this. Nevertheless, some
         relay agent implementations incorrectly relay such messages.

   The consistency checks specified in Section 2.1 SHOULD be performed
   by the relay agent. BOOTP messages not meeting these consistency
   checks MUST be silently discarded.

4.1.1 BOOTREQUEST Messages

   Some configuration mechanism MUST exist to enable or disable the
   relaying of BOOTREQUEST messages. Relaying MUST be disabled by
   default.

When the BOOTP relay agent receives a BOOTREQUEST message, it MAY use
the value of the 'secs' (seconds since client began booting) field of
the request as a factor in deciding whether to relay the request. If
such a policy mechanism is implemented, its threshold SHOULD be
configurable.

   DISCUSSION:

      To date, this feature of the BOOTP protocol has not necessarily
      been shown to be useful. See Section 3.2 for a discussion.

The relay agent MUST silently discard BOOTREQUEST messages whose
'hops' field exceeds the value 16. A configuration option SHOULD be
provided to set this threshold to a smaller value if desired by the
network manager. The default setting for a configurable threshold
SHOULD be 4.

If the relay agent does decide to relay the request, it MUST examine
the 'giaddr' ("gateway" IP address) field. If this field is zero,
the relay agent MUST fill this field with the IP address of the
interface on which the request was received. If the interface has
more than one IP address logically associated with it, the relay
agent SHOULD choose one IP address associated with that interface and
use it consistently for all BOOTP messages it relays. If the
'giaddr' field contains some non-zero value, the 'giaddr' field MUST
NOT be modified. The relay agent MUST NOT, under any circumstances,
fill the 'giaddr' field with a broadcast address as is suggested in
[1] (Section 8, sixth paragraph).

The value of the 'hops' field MUST be incremented.

All other BOOTP fields MUST be preserved intact.

At this point, the request is relayed to its new destination (or
destinations). This destination MUST be configurable. Further, this
destination configuration SHOULD be independent of the destination
configuration for any other so-called "broadcast forwarders" (e.g.,
for the UDP-based TFTP, DNS, Time, etc. protocols).

   DISCUSSION:

      The network manager may wish the relaying destination to be an
      IP unicast, multicast, broadcast, or some combination. A
      configurable list of destination IP addresses provides good
      flexibility. More flexible configuration schemes are
      encouraged. For example, it may be desirable to send to the
      limited broadcast address (255.255.255.255) on specific
      physical interfaces. However, if the BOOTREQUEST message was

received as a broadcast, the relay agent MUST NOT rebroadcast the BOOTREQUEST on the physical interface from whence it came.

A relay agent MUST use the same destination (or set of destinations) for all BOOTREQUEST messages it relays from a given client.

DISCUSSION:

At least one known relay agent implementation uses a round-robin scheme to provide load balancing across multiple BOOTP servers. Each time it receives a new BOOTREQUEST message, it relays the message to the next BOOTP server in a list of servers. Thus, with this relay agent, multiple consecutive BOOTREQUEST messages from a given client will be delivered to different servers.

Unfortunately, this well-intentioned scheme reacts badly with DHCP [3] and perhaps other variations of the BOOTP protocol which depend on multiple exchanges of BOOTREQUEST and BOOTREPLY messages between clients and servers. Therefore, all BOOTREQUEST messages from a given client MUST be relayed to the same destination (or set of destinations).

One way to meet this requirement while providing some load-balancing benefit is to hash the client's link-layer address (or some other reliable client-identifying information) and use the resulting hash value to select the appropriate relay destination (or set of destinations). The simplest solution, of course, is to not use a load-balancing scheme and just relay ALL received BOOTREQUEST messages to the same destination (or set of destinations).

When transmitting the request to its next destination, the relay agent may set the IP Time-To-Live field to either the default value for new datagrams originated by the relay agent, or to the TTL of the original BOOTREQUEST decremented by (at least) one.

DISCUSSION:

As an extra precaution against BOOTREQUEST loops, it is preferable to use the decremented TTL from the original BOOTREQUEST. Unfortunately, this may be difficult to do in some implementations.

If the BOOTREQUEST has a UDP checksum (i.e., the UDP checksum is non-zero), the checksum must be recalculated before

transmitting the request.

4.1.2 BOOTREPLY Messages

    BOOTP relay agents relay BOOTREPLY messages only to BOOTP clients.
    It is the responsibility of BOOTP servers to send BOOTREPLY messages
    directly to the relay agent identified in the 'giaddr' field.
    Therefore, a relay agent may assume that all BOOTREPLY messages it
    receives are intended for BOOTP clients on its directly-connected
    networks.

    When a relay agent receives a BOOTREPLY message, it should examine
    the BOOTP 'giaddr', 'yiaddr', 'chaddr', 'htype', and 'hlen' fields.
    These fields should provide adequate information for the relay agent
    to deliver the BOOTREPLY message to the client.

    The 'giaddr' field can be used to identify the logical interface from
    which the reply must be sent (i.e., the host or router interface
    connected to the same network as the BOOTP client). If the content
    of the 'giaddr' field does not match one of the relay agent's
    directly-connected logical interfaces, the BOOTREPLY messsage MUST be
    silently discarded.

    The 'htype', 'hlen', and 'chaddr' fields supply the link-layer
    hardware type, hardware address length, and hardware address of the
    client as defined in the ARP protocol [4] and the Assigned Numbers
    document [6]. The 'yiaddr' field is the IP address of the client, as
    assigned by the BOOTP server.

    The relay agent SHOULD examine the newly-defined BROADCAST flag (see
    Sections 2.2 and 3.1.1 for more information). If this flag is set to
    1, the reply SHOULD be sent as an IP broadcast using the IP limited
    broadcast address 255.255.255.255 as the IP destination address and
    the link-layer broadcast address as the link-layer destination
    address. If the BROADCAST flag is cleared (0), the reply SHOULD be
    sent as an IP unicast to the IP address specified by the 'yiaddr'
    field and the link-layer address specified in the 'chaddr' field. If
    unicasting is not possible, the reply MAY be sent as a broadcast, in
    which case it SHOULD be sent to the link-layer broadcast address
    using the IP limited broadcast address 255.255.255.255 as the IP
    destination address.

        DISCUSSION:

            The addition of the BROADCAST flag to the protocol is a
            workaround to help promote interoperability with certain client
            implementations.

Note that since the 'flags' field was previously defined in [1]
simply as an "unused" field, it is possible that old client or
server implementations may accidentally and unknowingly set the
new BROADCAST flag. It is actually expected that such
implementations will be rare (most implementations seem to
zero-out this field), but interactions with such
implementations must nevertheless be considered. If an old
client or server does set the BROADCAST flag to 1 incorrectly,
conforming relay agents will generate broadcast BOOTREPLY
messages to the corresponding client. The BOOTREPLY messages
should still properly reach the client, at the cost of one
(otherwise unnecessary) additional broadcast. This, however,
is no worse than a server or relay agent which always
broadcasts its BOOTREPLY messages.

Older client or server implementations which accidentally set
the BROADCAST flag SHOULD be corrected to properly comply with
this newer specification.

All BOOTP fields MUST be preserved intact. The relay agent
MUST NOT modify any BOOTP field of the BOOTREPLY message when
relaying it to the client.

The reply MUST have its UDP destination port set to BOOTPC
(68).

If the BOOTREPLY has a UDP checksum (i.e., the UDP checksum is
non-zero), the checksum must be recalculated before
transmitting the reply.

5. BOOTP Server Behavior

   This section provides clarifications on the behavior of BOOTP
   servers.

5.1 Reception of BOOTREQUEST Messages

   All received UDP messages whose UDP destination port number is BOOTPS
   (67) are considered for processing by the BOOTP server.

   Hosts and routers are usually required to silently discard incoming
   datagrams containing illegal IP source addresses. This is generally
   known as "Martian address filtering." One of these illegal addresses
   is 0.0.0.0 (or actually anything on network 0). However, hosts or
   routers which support a BOOTP server MUST accept for local delivery
   to the server BOOTREQUEST messages whose IP source address is
   0.0.0.0. BOOTREQUEST messages from legal IP source addresses MUST
   also be accepted.

A BOOTP server MUST silently discard any received UDP messages whose UDP destination port number is BOOTPC (68).

    DISCUSSION:

        There should be no need for a BOOTP server to process messages addressed to the BOOTPC port. Careful reading of the original BOOTP specification [1] will show this.

        The consistency checks specified in Section 2.1 SHOULD be performed by the BOOTP server. BOOTP messages not meeting these consistency checks MUST be silently discarded.

5.2 Use of the 'secs' field

When the BOOTP server receives a BOOTREQUEST message, it MAY use the value of the 'secs' (seconds since client began booting) field of the request as a factor in deciding whether and/or how to reply to the request.

    DISCUSSION:

        To date, this feature of the BOOTP protocol has not necessarily been shown to be useful. See Section 3.2 for a discussion.

5.3 Use of the 'ciaddr' field

There have been various client interpretations of the 'ciaddr' field for which Section 3.3 should be consulted. A BOOTP server SHOULD be prepared to deal with these varying interpretations. In general, the 'ciaddr' field SHOULD NOT be trusted as a sole key in identifying a client; the contents of the 'ciaddr', 'chaddr', 'htype', and 'hlen' fields, and probably other information (perhaps in the 'file' and 'vend' fields) SHOULD all be considered together in deciding how to respond to a given client.

BOOTP servers SHOULD preserve the contents of the 'ciaddr' field in BOOTREPLY messages; the contents of 'ciaddr' in a BOOTREPLY message SHOULD exactly match the contents of 'ciaddr' in the corresponding BOOTREQUEST message.

    DISCUSSION:

        It has been suggested that a client may wish to use the contents of 'ciaddr' to further verify that a particular BOOTREPLY message was indeed intended for it.

5.4 Strategy for Delivery of BOOTREPLY Messages

   Once the BOOTP server has created an appropriate BOOTREPLY message,
   that BOOTREPLY message must be properly delivered to the client.

   The server SHOULD first check the 'ciaddr' field. If the 'ciaddr'
   field is non-zero, the BOOTREPLY message SHOULD be sent as an IP
   unicast to the IP address identified in the 'ciaddr' field. The UDP
   destination port MUST be set to BOOTPC (68). However, the server
   MUST be aware of the problems identified in Section 3.3. The server
   MAY choose to ignore the 'ciaddr' field and act as if the 'ciaddr'
   field contains 0.0.0.0 (and thus continue with the rest of the
   delivery algorithm below).

   The server SHOULD next check the 'giaddr' field. If this field is
   non-zero, the server SHOULD send the BOOTREPLY as an IP unicast to
   the IP address identified in the 'giaddr' field. The UDP destination
   port MUST be set to BOOTPS (67). This action will deliver the
   BOOTREPLY message directly to the BOOTP relay agent closest to the
   client; the relay agent will then perform the final delivery to the
   client. If the BOOTP server has prior knowledge that a particular
   client cannot receive unicast BOOTREPLY messages (e.g., the network
   manager has explicitly configured the server with such knowledge),
   the server MAY set the newly-defined BROADCAST flag to indicate that
   relay agents SHOULD broadcast the BOOTREPLY message to the client.
   Otherwise, the server MUST preserve the state of the BROADCAST flag
   so that the relay agent can correctly act upon it.

   If the 'giaddr' field is set to 0.0.0.0, then the client resides on
   one of the same networks as the BOOTP server. The server SHOULD
   examine the newly-defined BROADCAST flag (see Sections 2.2, 3.1.1 and
   4.1.2 for more information). If this flag is set to 1 or the server
   has prior knowledge that the client is unable to receive unicast
   BOOTREPLY messages, the reply SHOULD be sent as an IP broadcast using
   the IP limited broadcast address 255.255.255.255 as the IP
   destination address and the link-layer broadcast address as the
   link-layer destination address. If the BROADCAST flag is cleared
   (0), the reply SHOULD be sent as an IP unicast to the IP address
   specified by the 'yiaddr' field and the link-layer address specified
   in the 'chaddr' field. If unicasting is not possible, the reply MAY
   be sent as a broadcast in which case it SHOULD be sent to the link-
   layer broadcast address using the IP limited broadcast address
   255.255.255.255 as the IP destination address. In any case, the UDP
   destination port MUST be set to BOOTPC (68).

     DISCUSSION:

          The addition of the BROADCAST flag to the protocol is a
          workaround to help promote interoperability with certain client
          implementations.

          The following table summarizes server delivery decisions for
          BOOTREPLY messages based upon information in BOOTREQUEST
          messages:

       BOOTREQUEST fields   BOOTREPLY values for UDP, IP, link-layer
     +----------------------+---------------------------------------+
     | 'ciaddr' 'giaddr' B | UDP dest   IP destination   link dest |
     +----------------------+---------------------------------------+
     | non-zero   X   X | BOOTPC (68) 'ciaddr'   normal |
     | 0.0.0.0   non-zero X | BOOTPS (67) 'giaddr'   normal |
     | 0.0.0.0   0.0.0.0   0 | BOOTPC (68) 'yiaddr'   'chaddr' |
     | 0.0.0.0   0.0.0.0   1 | BOOTPC (68) 255.255.255.255   broadcast |
     +----------------------+---------------------------------------+

          B = BROADCAST flag

          X = Don't care

     normal = determine from the given IP destination using normal
              IP routing mechanisms and/or ARP as for any other
              normal datagram

Acknowledgements

   The author would like to thank Gary Malkin for his contribution of
   the "BOOTP over IEEE 802.5 Token Ring Networks" section, and Steve
   Deering for his observations on the problems associated with the
   'giaddr' field.

   Ralph Droms and the many members of the IETF Dynamic Host
   Configuration and Router Requirements working groups provided ideas
   for this memo as well as encouragement to write it.

   Philip Almquist and David Piscitello offered many helpful suggestions
   for improving the clarity, accuracy, and organization of this memo.
   These contributions are graciously acknowledged.

References

   [1] Croft, B., and J. Gilmore, "Bootstrap Protocol (BOOTP)", RFC 951,
       Stanford University and Sun Microsystems, September 1985.

   [2] Reynolds, J., "BOOTP Vendor Information Extensions", RFC 1497,
       USC/Information Sciences Institute, August 1993. This RFC is
       occasionally reissued with a new number. Please be sure to
       consult the latest version.

   [3] Droms, R., "Dynamic Host Configuration Protocol", RFC 1541,
       Bucknell University, October 1993.

   [4] Plummer, D., "An Ethernet Address Resolution Protocol", STD 37,
       RFC 826, MIT, November 1982.

   [5] Deering, S., "ICMP Router Discovery Messages", RFC 1256, Xerox
       PARC, September 1991.

   [6] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1340,
       USC/Information Sciences Institute, July, 1992. This RFC is
       periodically reissued with a new number. Please be sure to
       consult the latest version.

Security Considerations

   There are many factors which make BOOTP in its current form quite
   insecure. BOOTP is built directly upon UDP and IP which are as yet
   inherently insecure themselves. Furthermore, BOOTP is generally
   intended to make maintenance of remote and/or diskless hosts easier.
   While perhaps not impossible, configuring such hosts with passwords or
   keys may be difficult and inconvenient. This makes it difficult to
   provide any form of reasonable authentication between servers and
   clients.

   Unauthorized BOOTP servers may easily be set up. Such servers can
   then send false and potentially disruptive information to clients such
   as incorrect or duplicate IP addresses, incorrect routing information
   (including spoof routers, etc.), incorrect domain nameserver addresses
   (such as spoof nameservers), and so on. Clearly, once this "seed"
   mis-information is planted, an attacker can further compromise the
   affected systems.

   Unauthorized BOOTP relay agents may present some of the same problems
   as unauthorized BOOTP servers.

   Malicious BOOTP clients could masquerade as legitimate clients and
   retrieve information intended for those legitimate clients. Where
   dynamic allocation of resources is used, a malicious client could
   claim all resources for itself, thereby denying resources to
   legitimate clients.

Author's Address

   Walt Wimer
   Network Development
   Carnegie Mellon University
   5000 Forbes Avenue
   Pittsburgh, PA 15213-3890

   Phone: (412) 268-6252
   EMail: Walter.Wimer@CMU.EDU