Internet Engineering Task Force
Internet Draft

W. Wimer
Carnegie Mellon University
September, 1992

DRAFT

Clarifications and Extensions for the Bootstrap Protocol

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. (Note that other groups may also distribute working documents as Internet Drafts.)

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft. This Internet Draft expires on February 28, 1993.

This memo suggests several updates to the specification of the Bootstrap Protocol (BOOTP) based on experience with the protocol.

This proposal is a product of the IETF Dynamic Host Configuration Working Group. This draft document will be submitted to the RFC editor as a protocol specification. Comments on this memo should be sent to the IETF Dynamic Host Configuration Working Group mailing list, host-conf@sol.bucknell.edu.

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.

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 [2,3].

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.

1.1 Requirements

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

- "MUST"

  This word or the adjective "REQUIRED" means that the item

is an absolute requirement of the specification.

- "MUST NOT"

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

- "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.

- "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.

- "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:

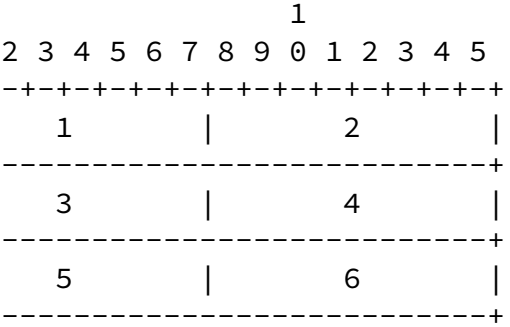
BOOTREQUEST
: A BOOTREQUEST message is a BOOTP message sent from a BOOTP client to a BOOTP server, requesting configuration information.

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

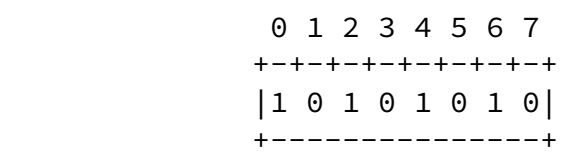
Silently discard
: This memo specifies several cases where a BOOTP relay agent is to "silently discard" a received BOOTP message. This means that the relay agent should discard the message without further processing, and that the relay agent will not send any ICMP error message as a result. However, for diagnosis of problems, the relay agent 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.



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).



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.

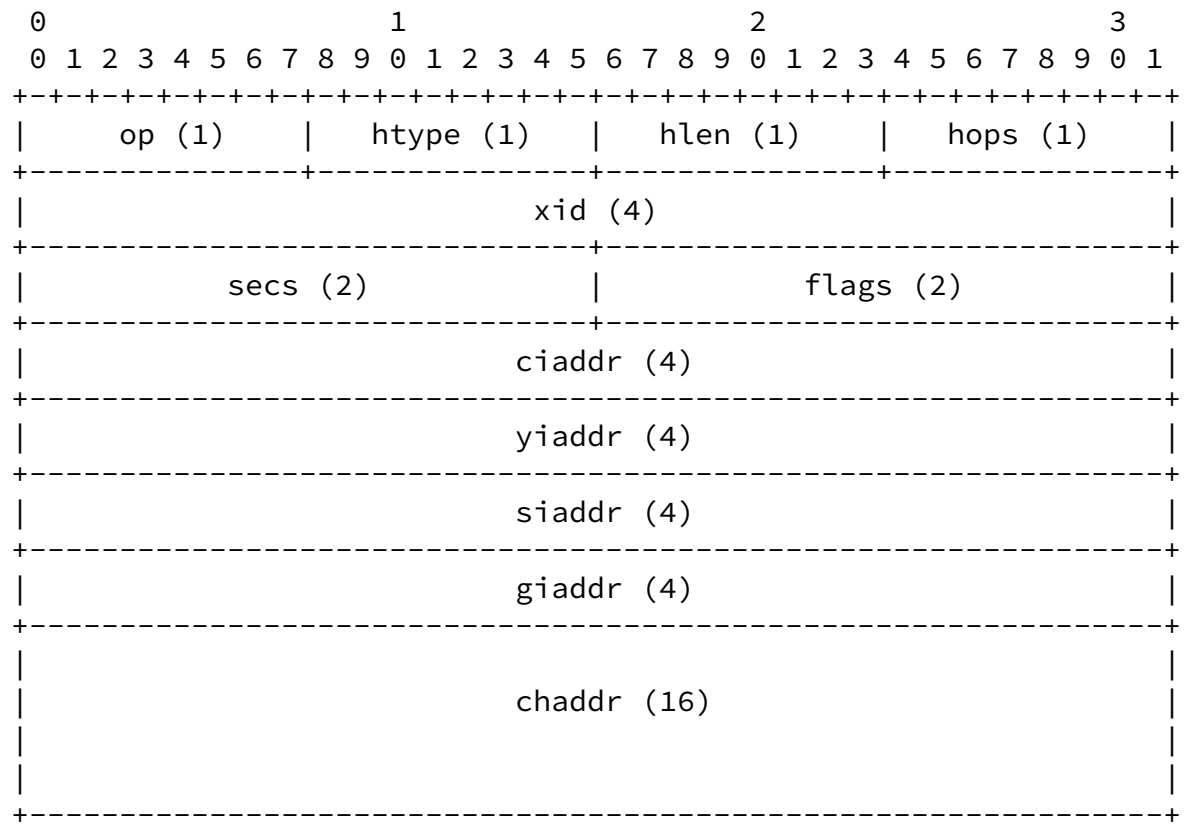
2. Definition of the 'flags' Field

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.

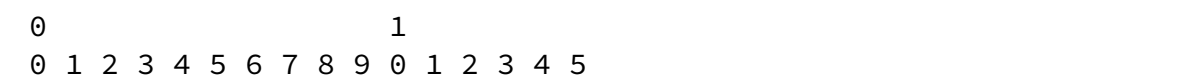
The first 44 octets of a BOOTP message are shown below. The numbers in parentheses indicate the size of each field in octets.

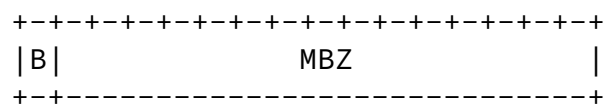


This document hereby defines the most significant bit of the 'flags' field as the BROADCAST (B) flag. The semantics of this flag are discussed in Sections 3.1.2 and 4.1.1 of this memo.

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:





where:

B BROADCAST flag (discussed in Sections 3.1.2 and 4.1.1)

MBZ MUST BE ZERO (reserved for future use)

3. 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. One should resist the notion of simply forwarding 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.

3.1 General BOOTP Processing

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, local delivery has a similar but somewhat more careful definition for which [4] should be consulted.

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, of course.

The following consistency checks SHOULD be performed on BOOTP messages:

- 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.

- 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.

3.1.1 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 4.2 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 logical interface on which the request was received. In addition, the relay agent MAY insert the subnet mask of that logical interface into the vendor area (see the next paragraph for details). If the 'giaddr' field contains some non-zero value, the 'giaddr' field MUST NOT be modified and the subnet mask MUST NOT be inserted into the vendor area nor 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).

To insert the subnet mask into the vendor area as suggested above, the relay agent MUST examine the first four octets of the 'vend' field (these first four octets are usually referred to as the "vendor magic number" or "vendor magic cookie"). If these four octets do not contain the dotted decimal value 99.130.83.99 as specified in [2,3], the subnet mask MUST NOT be inserted. If these four octets do contain the value 99.130.83.99, it is safe to insert the subnet mask. The relay agent MUST use the data

format as specified in [2,3] and MUST use the "Subnet Mask Field" (Tag 1) specified in [2,3] to express the subnet mask. The relay agent MUST be careful to preserve any and all existing data in the 'vend' field. The subnet mask MUST either be placed at the beginning of the data portion of the 'vend' field (immediately after the four-octet magic cookie), or the relay agent MUST be careful to replace any existing subnet mask entries (Tag 1) with the correct subnet mask value. This is to

avoid any ambiguity in the event that the client supplied one or more subnet mask entries somewhere in the 'vend' field. If the subnet mask cannot be inserted without loss of data in the 'vend' field, the subnet mask MUST NOT be inserted.

DISCUSSION:
Having the relay agent insert the subnet mask into the vendor area is an optimization for the proposed Dynamic Host Configuration Protocol (DHCP) [5]. This optimization should not be strictly necessary for correct operation of the protocol, but it should make configuration of the DHCP server much easier. It is strongly encouraged that relay agents provide this subnet mask feature, but it is not absolutely required.

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

All other 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 logical interfaces. However, if the BOOTREQUEST message was received as a broadcast, the relay agent MUST NOT rebroadcast the BOOTREQUEST on the logical 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 certain 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.

The UDP checksum must be recalculated before transmitting the request.

3.1.2 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 to 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 [6] and the Assigned Numbers document [7]. 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 and 4.1.1 for more information). If this flag is set to 1, the reply SHOULD be sent as an IP broadcast using an IP broadcast address (preferably 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 to the link-layer broadcast address using an IP broadcast address (preferably 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.

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

The UDP checksum must be recalculated before transmitting the reply.

4. BOOTP Client Behavior

This section clarifies various issues regarding BOOTP client behavior.

4.1 Client use of the 'flags' field

4.1.1 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. It is strongly recommended that such implementations 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.

4.1.2 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.

4.2 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 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 (usually resulting in very inflated figures).

4.3 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 consider the 'giaddr' field of a BOOTREPLY

message to represent 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 3.1.1 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" [2,3] (if present), or from some other router discovery protocol.

4.4 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 [2,3]. 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 BOOTREQUEST messages, the server will at least know what format the client expects and can understand in corresponding BOOTREPLY messages.

5. 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 [6] on the client's network (assuming ARP is defined for that network).

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.

6. BOOTP Over IEEE 802.5 Token Ring Networks

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

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 is 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.

Security Considerations

BOOTP is built directly upon UDP and IP which are as yet inherently insecure. 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. Therefore, BOOTP in its current form is quite insecure.

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.

BOOTP relay agents suffer some of the same problems as 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.

Acknowledgements

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

Ralph Droms of Bucknell University 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.

References

[1] Croft, B., and J. Gilmore. Bootstrap Protocol (BOOTP). Request For Comments (RFC) 951, DDN Network Information Center, SRI International, Menlo Park, California, USA, September, 1985.

[2] Prindeville, P. BOOTP Vendor Information Extensions. Request For Comments (RFC) 1048, DDN Network Information Center, SRI International, Menlo Park, California, USA, February, 1988.

[3] Reynolds, J. BOOTP Vendor Information Extensions. Request For Comments (RFC) 1084, DDN Network Information Center, SRI International, Menlo Park, California, USA, December, 1988.

[4] Almquist, P. Requirements for IP Routers. Internet Draft, Corporation for National Research Initiatives, Reston, Virginia, USA, May, 1991.

[5] Droms, R. Dynamic Host Configuration Protocol. Internet Draft, Corporation for National Research Initiatives, Reston, Virginia, USA, June, 1992.

[6] Plummer, D. An Ethernet Address Resolution Protocol. Request For Comments (RFC) 826, DDN Network Information Center, SRI International, Menlo Park, California, USA, November, 1982.

[7] Reynolds, J., and J. Postel. Assigned Numbers. Request For Comments (RFC) 1340, DDN Network Information Center, SRI International, Menlo Park, California, USA, July, 1992. This RFC is periodically reissued with a new number. Please be sure to consult the latest version.

Author's Address

Walt Wimer
Network Development
Carnegie Mellon University
4910 Forbes Avenue
Pittsburgh, PA 15213-3890

Phone: (412) 268-6252

EMail: Walter.Wimer@ANDREW.CMU.EDU