

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

The process of starting a computer system is known as *bootstrapping*. In most systems, the initial bootstrap sequence begins with code in ROM, which the CPU executes. The ROM code only contains a first step—it merely loads an image into the computer's RAM and branches to the image. There are two approaches used to obtain an image:

- **Embedded system:** On a diskless computer, the ROM code contains sufficient support software to permit network communication. The ROM code uses the network support to locate and download an image.
- **Conventional computer:** On a computer that has secondary storage (for instance, a PC), the ROM code loads the image from a well-known place on disk. Typically, the loaded image consists of an operating system that then controls the computer.

In either case, the image loaded by ROM is not tailored to the specific physical hardware. Instead, an image is *generic*, which means that before it can be used, it must be configured for the local hardware. In particular, the image does not contain such networking details as the computer's IP address, address mask, or domain name. Each of these items must be supplied before applications can use TCP/IP.

Early in the history of TCP/IP, designers chose to provide a separate mechanism for each item of configuration information. Thus, the *Reverse Address Resolution Protocol* (RARP) only allowed a computer to obtain its IP address. When subnet masks were introduced, ICMP Address Mask messages were added to allow a computer to obtain a subnet mask. The chief advantage of such an approach lies in flexibility—a computer can decide which items to obtain from a local file on disk and which to obtain over the network. The chief disadvantage becomes apparent when one considers the network traffic and delay. A given computer must issue a series of small request messages. More important, each response returns a small value (for instance, a 4-octet IP address). Because networks enforce a minimum packet size, most of the space in each packet is wasted.

BOOTP

As the complexity of configuration grew, TCP/IP protocol designers observed that many of the configuration steps could be combined into a single step if a server was able to supply more than one item of configuration information. To provide such a service, the designers invented the *BOOTstrap Protocol* (BOOTP). To obtain configuration information, protocol software broadcasts a *BOOTP Request* message.

A BOOTP server that receives the request looks up several pieces of configuration information for the computer that issued the request, places the information in a single BOOTP Response message, and returns the reply to the requesting computer. Thus, in a single step, a computer can obtain information such as the computer's IP address, the server's name and IP address, and the IP address of a default router.

Like other protocols used to obtain configuration information, BOOTP broadcasts each request. Unlike other protocols used for configuration, BOOTP appears to use a protocol that has not been configured: BOOTP uses IP to send a request and receive a response. How can BOOTP send an IP datagram before a computer's IP address has been configured? The answer lies in a careful design

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

that allows IP to broadcast a request and receive a response before all values have been configured. To send a BOOTP datagram, IP uses the all-1's limited broadcast address as a DESTINATION ADDRESS , and uses the all-0's address as a SOURCE ADDRESS . If a computer uses the all-0's address to send a request, a BOOTP server either uses broadcast to return the response or uses the hardware address on the incoming frame to send a response via unicast. (The server must be careful to avoid using ARP because a client that does not know its IP address cannot answer ARP requests.)

Thus, a computer that does not know its IP address can communicate with a BOOTP server. Figure 1 illustrates the BOOTP packet format. The message is sent using UDP, which is encapsulated in IP.

Figure 1: BOOTP Packet Format

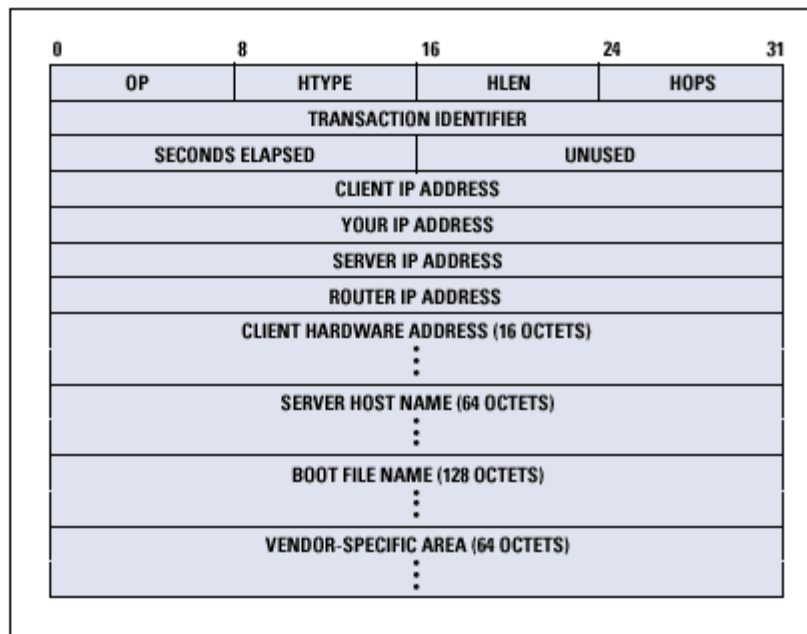


Figure 1: BOOTP Packet Format

Each field in a BOOTP message has a fixed size. The first seven fields contain information used to process the message. The OP field specifies whether the message is a Request or a Response , and the HTYPE and HLEN fields specify the network hardware type and the length of a hardware address. The HOPS field specifies how many servers forwarded the request, and the TRANSACTION IDENTIFIER field provides a value that a client can use to determine if an incoming response matches its request. The SECONDS ELAPSED field specifies how many seconds have elapsed since the computer began to boot. Finally, if a computer knows its IP address (for instance, the address was obtained using RARP), the computer fills in the CLIENT IP ADDRESS field in a request.

Later fields are used in a response message to carry information back to the computer that is booting. If a computer does not know its address, the server uses field YOUR IP ADDRESS to supply the value. In addition, the server uses fields SERVER IP ADDRESS and SERVER HOST NAME to give

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

the computer information about the location of a computer that runs servers. Field ROUTER IP ADDRESS contains the IP address of a default router.

In addition to protocol configuration, BOOTP allows a computer to negotiate to find a boot image. To do so, the computer fills in field BOOT FILE NAME with a generic request (for instance, the computer can request the UNIX operating system). The BOOTP server does not send an image. Instead, the server determines which file contains the requested image, and uses field BOOT FILE NAME to send back the name of the file. Once a BOOTP response arrives, a computer must use a protocol like the Trivial File Transfer Protocol (TFTP) to obtain a copy of the image.

Automatic Address Assignment

Although it simplifies loading parameters into protocol software, BOOTP does not solve the configuration problem completely. When a BOOTP server receives a request, the server looks up the computer in its database of information. Thus, even a computer that uses BOOTP cannot boot on a new network until the administrator manually changes information in the database.

Can protocol software be devised that allows a computer to join a new network without manual intervention? Yes—several such protocols exist. For example, IPX and IPv6 can generate a protocol address from the computer's hardware address. To make automatic generation work correctly, the hardware address must be unique. Furthermore, if the hardware address and protocol address are not the same size, it must be possible to translate the hardware address into a protocol address that is also unique.

The AppleTalk protocols use a bidding scheme to allow a computer to join a new network. When a computer first boots, the computer chooses a random address. For example, suppose computer C chooses address 17. To ensure that no other computer on the network is using the address, C broadcasts a request message and starts a timer. If no other computer is using address 17, no reply will arrive before the timer expires; C can begin using address 17. If another computer is using 17, the computer replies, causing C to choose a different address and begin again.

Choosing an address at random works well for small networks and for computers that run client software. However, the scheme does not work well for servers. To understand why, recall that each server must be located at a well-known address. If a computer chooses an address at random when it boots, clients will not know which address to use when contacting a server on that computer. More important, because the address can change each time a computer boots, the address used to reach a server may not remain the same after a crash and reboot.

A bidding scheme also has the disadvantage that two computers can choose the same network address. In particular, assume that computer B sends a request for an address that another computer (for example, A) is already using. If A fails to respond to the request for any reason, both computers will attempt to use the same address, with disastrous results. In practice, such failures can occur for a variety of reasons. For example, a piece of network equipment such as a bridge can fail, a computer can be unplugged from the network when the request is sent, or a computer can be temporarily unavailable (for instance, in a hibernation mode designed to conserve power). Finally, a computer can fail to answer if the protocol software or operating system is not functioning correctly.

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

DHCP

To automate configuration, the Internet Engineering Task Force (IETF) devised the Dynamic Host Configuration Protocol (DHCP). Unlike BOOTP, DHCP does not require an administrator to add an entry for each computer to the database that a server uses. Instead, DHCP provides a mechanism that allows a computer to join a new network and obtain an IP address without manual intervention. The concept has been termed plug-and-play networking. More important, DHCP accommodates computers that run server software as well as computers that run client software:

- When a computer that runs client software is moved to a new network, the computer can use DHCP to obtain configuration information without manual intervention.
- DHCP allows nonmobile computers that run server software to be assigned a permanent address; the address will not change when the computer reboots.

To accommodate both types of computers, DHCP cannot use a bidding scheme. Instead, it uses a client-server approach. When a computer boots, the computer broadcasts a DHCP Request to which a server sends a DHCP Reply. (The reply is classified as a DHCP offer message that contains an address the server is offering to the client.)

An administrator can configure a DHCP server to have two types of addresses: permanent addresses that are assigned to server computers, and a pool of addresses to be allocated on demand. When a computer boots and sends a request to DHCP, the DHCP server consults its database to find configuration information.

If the database contains a specific entry for the computer, the server returns the information from the entry. If no entry exists for the computer, the server chooses the next IP address from the pool, and assigns the address to the computer.

In fact, addresses assigned on demand are not permanent. Instead, DHCP issues a lease on the address for a finite period of time. (When the administrator establishes a pool of addresses for DHCP to assign, the administrator must also specify the length of the lease for each address.)

When the lease expires, the computer must renegotiate with DHCP to extend the lease. Normally, DHCP will approve a lease extension. However, a site may choose an administrative policy that denies the extension. (For example, a university that has a network in a classroom might choose to deny extensions on leases at the end of a class period to allow the next class to reuse the same addresses.) If DHCP denies an extension request, the computer must stop using the address.

Optimizations in DHCP

If the computers on a network use DHCP to obtain configuration information when they boot, an event that causes all computers to restart at the same time can cause the network or server to be flooded with requests. To avoid the problem, DHCP uses the same technique as BOOTP: each computer waits a random time before transmitting or retransmitting a request.

The DHCP protocol has two steps: one in which a computer broadcasts a DHCP Discover message

One Byte at a Time: Bootstrapping with BOOTP & DHCP

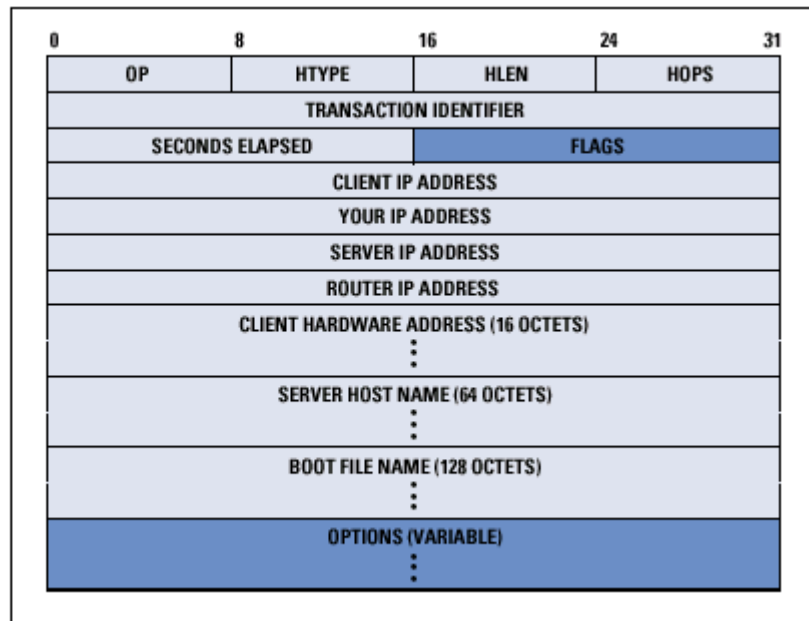
Douglas Comer

to find a DHCP server, and another in which the computer selects one of the servers that responded to its message and sends a request to that server. To avoid having a computer repeat both steps each time it boots or each time it needs to extend the lease, DHCP uses caching. When a computer discovers a DHCP server, the computer saves the server's address in a cache on permanent storage (for example, a disk file). Similarly, once it obtains an IP address, the computer saves the IP address in a cache. When a computer reboots, it uses the cached information to revalidate its former address. Doing so saves time and reduces network traffic.

DHCP Message Format

Interestingly, DHCP is designed as an extension of BOOTP. As Figure 2 illustrates, DHCP uses a slightly modified version of the BOOTP message format.

Figure 2: DHCP Message Format



Most of the fields in a DHCP message have the same meaning as in BOOTP; DHCP replaces the 16-bit *UNUSED* field with a *FLAGS* field, and uses the *OPTIONS* field to encode additional information. For example, as in BOOTP, the *OP* field specifies either a *Request* or a *Response*. To distinguish among various messages that a client uses to discover servers or request an address, or that a server uses to acknowledge or deny a request, DHCP uses a *message type option*. That is, each message contains a code that identifies the message type.

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

DHCP and Domain Names

Although DHCP makes it possible for a computer to obtain an IP address without manual intervention, DHCP does not interact with the Domain Name System. As a result, a computer cannot keep its name when it changes addresses. Interestingly, the computer does not need to move to a new network to have its name change. For example, suppose a computer obtains IP address 192.5.48.195 from DHCP, and suppose the domain name system contains a record that binds the name x.y.z.com to the address. Now consider what happens if the owner turns off the computer and takes a two-month vacation during which the address lease expires. DHCP may assign the address to another computer. When the owner returns and turns on the computer, DHCP will deny the request to use the same address. Thus, the computer will obtain a new address. Unfortunately, the *Domain Name System* (DNS) continues to map the name x.y.z.com to the old address.

For several years, researchers have been considering how DHCP should interact with the DNS. Although a dynamic DNS update protocol has been defined, it has not been widely deployed. Thus, many sites that use DHCP do not have a mechanism to update a DNS database. From a user's perspective, the lack of communication between DHCP and DNS means that when a computer is assigned a new address, the computer's name changes.

Summary

The bootstrapping sequence loads a generic image into a computer, either from secondary storage or over the network. Before application software can use TCP/IP protocols, the image must be configured by supplying values for internal parameters such as the IP address and subnet mask, and for external parameters such as the address of a default router; the process is known as *configuration*. Initially, separate protocols were used to obtain each piece of configuration information. Later, the *BOOTstrap Protocol*, BOOTP, was invented to consolidate separate requests into a single protocol. A BOOTP response provides information such as the computer's IP address, the address of a default router, and the name of a file that contains a boot image.

The *Dynamic Host Configuration Protocol* (DHCP) extends BOOTP. In addition to permanent addresses assigned to computers that run a server, DHCP permits completely automated address assignment. That is, DHCP allows a computer to join a new network, obtain a valid IP address, and begin using the address without requiring an administrator to enter information about the computer in a server's database. When DHCP allocates an address automatically, the DHCP server does not assign the address forever. Instead, the server specifies a lease during which the address may be used. A computer must extend the lease, or stop using the address when the lease expires.

For Further Study

Details about BOOTP can be found in reference [1], which compares BOOTP to RARP and serves as the official protocol standard. Reference [2] tells how to interpret the vendor-specific area, and reference [3] recommends using the vendor-specific area to pass the subnet mask. Most uses of BOOTP have been replaced by DHCP. Reference [4] contains the specification for DHCP, including a detailed description of state transitions. A related document, [5], specifies the encoding of DHCP options and BOOTP vendor extensions. Finally, reference [6] discusses the interoperability of BOOTP and DHCP. The chair of the DHCP working group, Ralph Droms, and Ted Lemon have written a book about DHCP [7].

One Byte at a Time: Bootstrapping with BOOTP & DHCP

Douglas Comer

References

- [1] W. J. Croft, J. Gilmore, "Bootstrap Protocol," RFC 951, September 1985.
- [2] J. K. Reynolds, "BOOTP Vendor Information Extensions," RFC 1084, December 1988.
- [3] R. Braden (ed), "Requirements for Internet Hosts—Application and Support," RFC 1123, October 1989.
- [4] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131, March 1997.
- [5] S. Alexander, R. Droms, "DHCP Options and BOOTP Vendor Extensions," RFC 2132, March 1997.
- [6] R. Droms, "Interoperation between DHCP and BOOTP," RFC 1534, October 1993.
- [7] R. Droms and T. Lemon, *The DHCP Handbook: Understanding, Deploying, and Managing Automated Configuration Services*, ISBN 1578701376, MacMillian, 1999.

[This article is adapted from *Computer Networks and Internets, with Internet Applications, 3rd edition*, by Douglas Comer, with CD by Ralph Droms, ISBN 0130914495, Prentice Hall, 2001.]

Dr. DOUGLAS COMER is a professor of Computer Science at Purdue University, consultant to industry, and an internationally recognized authority on TCP/IP. He has written numerous research papers and textbooks, including the classic three-volume reference series *Internetworking with TCP/IP*, and currently heads research projects. He designed and implemented X25NET and Cypress networks, and the Xinu operating system. He was a principal on the CSNET project, is director of the Internetworking Research Group at Purdue, editor of the journal *Software—Practice and Experience*, a former member of the IAB, and a Fellow of the ACM. E-mail: comer@cs.purdue.edu