CHAPTER

18

Host Configuration: DHCP

In this chapter we discuss our first client/server application program, Dynamic Host Configuration Protocol (DHCP). This application is discussed first because it is the first client/server application program that is used after a host is booted. In other words, it serves as a bootstrap when a host is booted and supposed to be connected to the Internet, but the host does not know its IP address.

OBJECTIVES

The chapter has several objectives:

- ❑ To give the reasons why we need host configuration.
- ❑ To give a historical background of two protocols used for host configuration in the past.
- ❑ To define DHCP as the current Dynamic Host Configuration Protocol.
- ❑ To discuss DHCP operation when the client and server are on the same network or on different networks.
- ❑ To show how DHCP uses two well-known ports of UDP to achieve configuration.
- ❑ To discuss the states the clients go through to lease an IP address from a DHCP server.

18.1 INTRODUCTION

Each computer that uses the TCP/IP protocol suite needs to know its IP address. If the computer uses classless addressing or is a member of a subnet, it also needs to know its subnet mask. Most computers today need two other pieces of information: the address of a default router to be able to communicate with other networks and the address of a name server to be able to use names instead of addresses as we will see in the next chapter. In other words, four pieces of information are normally needed:

1. The IP address of the computer
2. The subnet mask of the computer
3. The IP address of a router
4. The IP address of a name server

These four pieces of information can be stored in a configuration file and accessed by the computer during the bootstrap process. But what about a diskless workstation or a computer with a disk that is booted for the first time?

In the case of a diskless computer, the operating system and the networking software could be stored in read-only memory (ROM). However, the above information is not known to the manufacturer and thus cannot be stored in ROM. The information is dependent on the individual configuration of the machine and defines the network to which the machine is connected.

Previous Protocols

Before DHCP became the formal protocol for host configuration, some other protocols were used for this propose. We briefly describe them here.

RARP

At the beginning of the Internet era, a protocol called Reverse Address Resolution Protocol (RARP) was designed to provide the IP address for a booted computer. RARP was actually a version of ARP that we discussed in Chapter 8. ARP maps an IP address to a physical address: RARP maps a physical address to an IP address. However, RARP is deprecated today for two reasons. First, RARP used the broadcast service of the data link layer, which means that a RARP server must be present in each network. Second, RARP can provide only the IP address of the computer, but a computer today needs all four pieces of information mentioned above.

BOOTP

The **Bootstrap Protocol (BOOTP)** is the prerunner of DHCP. It is a client/server protocol designed to overcome the two deficiencies of the RARP protocol. First, since it is a client/server program, the BOOTP server can be anywhere in the Internet. Second, it can provide all pieces of information we mentioned above, including the IP address. To provide the four pieces of information described above, it removes all restriction about the RARP protocol. BOOTP, however, is a *static configuration protocol*. When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address. This implies that the binding between the physical address and the IP address of the client already exists. The binding is predetermined.

There are some situations in which we need a *dynamic configuration protocol*. For example, when a host moves from one physical network to another, its physical address changes. As another example, there are occasions when a host wants a temporary IP address to be used for a period of time. BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator. As we will see shortly, DHCP has been devised to handle these shortcomings.

DHCP

The **Dynamic Host Configuration Protocol (DHCP)** is a client/server protocol designed to provide the four pieces of information for a diskless computer or a computer that is booted for the first time. DHCP is a successor to BOOTP and is backward compatible with it. Although BOOTP is considered deprecated, there may be some systems that may still use BOOTP for host configuration. The part of the discussion in this chapter that does not deal with the dynamic aspect of DHCP can also be applied to BOOTP.

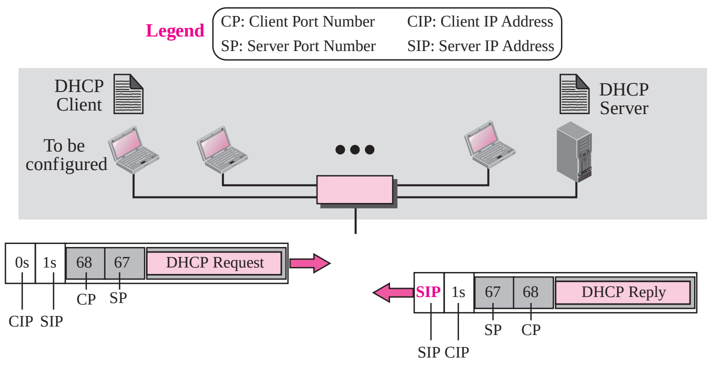
18.2 DHCP OPERATION

The DHCP client and server can either be on the same network or on different networks. Let us discuss each situation separately.

Same Network

Although the practice is not very common, the administrator may put the client and the server on the same network as shown in Figure 18.1.

Figure 18.1 *Client and server on the same network*



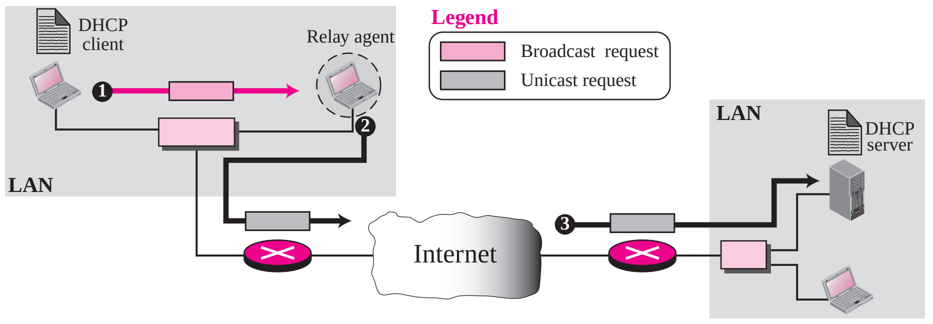
In this case, the operation can be described as follows:

1. The DHCP server issues a passive open command on UDP port number 67 and waits for a client.
2. A booted client issues an active open command on port number 68 (this number will be explained later). The message is encapsulated in a UDP user datagram, using the destination port number 67 and the source port number 68. The UDP user datagram, in turn, is encapsulated in an IP datagram. The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client uses all 0s as the source address and all 1s as the destination address.
3. The server responds with either a broadcast or a unicast message using UDP source port number 67 and destination port number 68. The response can be unicast because the server knows the IP address of the client. It also knows the physical address of the client, which means it does not need the services of ARP for logical to physical address mapping. However, some systems do not allow the bypassing of ARP, resulting in the use of the broadcast address.

Different Networks

As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks. Figure 18.2 shows the situation.

Figure 18.2 *Client and server on two different networks*



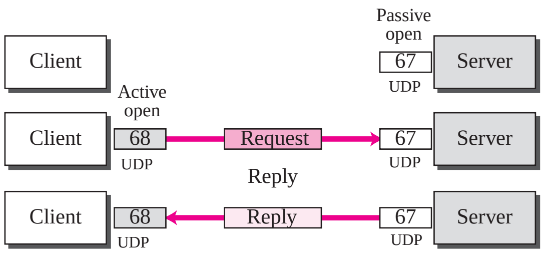
However, there is one problem that must be solved. The DHCP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. A router receiving such a packet discards it. Recall that an IP address of all 1s is a limited broadcast address.

To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this case is called a **relay agent.** The relay agent knows the unicast address of a DHCP server and listens for broadcast messages on port 67. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the DHCP server. The packet, carrying a unicast destination address, is routed by any router and reaches the DHCP server. The DHCP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the DHCP client.

UDP Ports

Figure 18.3 shows the interaction between a client and a DHCP server. The server uses the well-known port 67, which is normal. The client uses the well-known port 68, which is unusual. The reason for choosing the well-known port 68 instead of an ephemeral port is to prevent a problem when the reply, from the server to the client, is broadcast. To understand the problem, let us look at a situation where an ephemeral port is used. Suppose host A on a network is using a DHCP client on ephemeral port 2017 (randomly chosen). Host B, on the same network, is using a DAYTIME client on ephemeral port 2017 (accidentally the same). Now the DHCP server sends a broadcast reply message with the destination port number 2017 and broadcast IP address $FFFFFFFF_{16}$. Every host needs to open a packet carrying this destination IP address. Host A finds a message from an application program on ephemeral port 2017. A correct message is delivered to the DHCP client. An incorrect message is delivered to the DAYTIME client. The confusion is due to the demultiplexing of packets based on the socket address (see Chapter 17), which is a combination of IP address and port number. In this case, both are the same.

Figure 18.3 *Use of UDP ports*



The use of a well-known port (less than 1024) prevents the use of the same two destination port numbers. Host B cannot select 68 as the ephemeral port because ephemeral port numbers are greater than 1023.

The curious reader may ask what happens if host B is also running the DHCP client. In this case, the socket address is the same and both clients will receive the message. In this situation, a third identification number differentiates the clients. DHCP uses another number, called the transaction ID, which is randomly chosen for each connection involving DHCP. It is highly improbable that two hosts will choose the same ID at the same time.

Using TFTP

The server does not send all of the information that a client may need for booting. In the reply message, the server defines the pathname of a file in which the client can find complete booting information. The client can then use a TFTP message (see Chapter 21), which is encapsulated in a UDP user datagram, to obtain the rest of the needed information.

Error Control

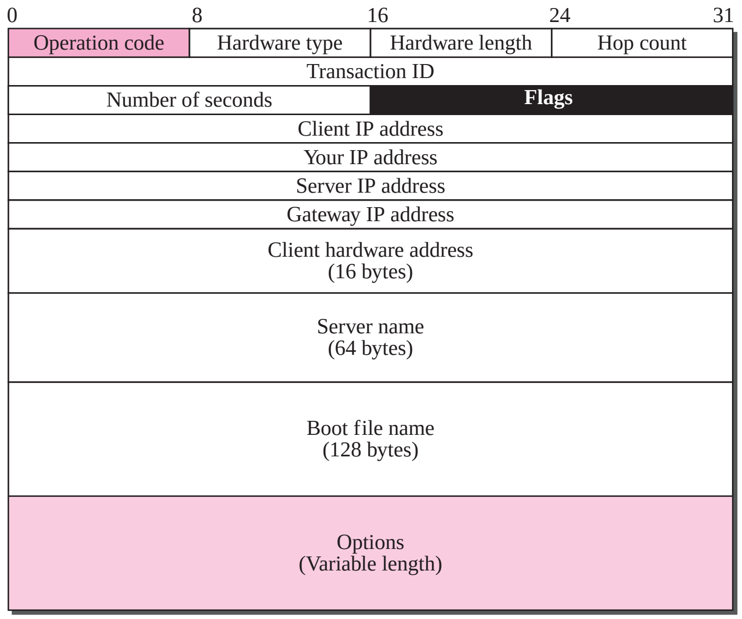
What if a request is lost or damaged? What if the response is damaged? There is a need for error control when using DHCP. DHCP uses UDP, which does not provide error control. Therefore, DHCP must provide error control. Error control is accomplished through two strategies:

1. DHCP requires that UDP uses the checksum. Remember that the use of the checksum in UDP is optional.
2. The DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request. However, to prevent a traffic jam when several hosts need to retransmit a request (for example, after a power failure), DHCP forces the client to use a random number to set its timers.

Packet Format

Figure 18.4 shows the format of a DHCP packet.

Figure 18.4 *DHCP packet format*

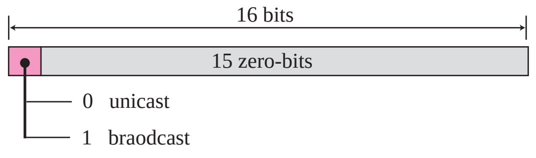


The following briefly describes each field:

- **Operation code.** This 8-bit field defines the type of DHCP packet: request (1) or reply (2).

- **Hardware type.** This is an 8-bit field defining the type of physical network. Each type of network has been assigned an integer. For example, for Ethernet the value is 1.
- **Hardware length.** This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.
- **Hop count.** This is an 8-bit field defining the maximum number of hops the packet can travel.
- **Transaction ID.** This is a 4-byte field carrying an integer. The transaction identification is set by the client and is used to match a reply with the request. The server returns the same value in its reply.
- **Number of seconds.** This is a 16-bit field that indicates the number of seconds elapsed since the time the client started to boot.
- **Flag.** This is a 16-bit field in which only the leftmost bit is used and the rest of the bits should be set to 0s. A leftmost bit specifies a forced broadcast reply (instead of unicast) from the server. If the reply were to be unicast to the client, the destination IP address of the IP packet is the address assigned to the client. Since the client does not know its IP address, it may discard the packet. However, if the IP datagram is broadcast, every host will receive and process the broadcast message. Figure 18.5 shows the flag format.

Figure 18.5 *Flag format*

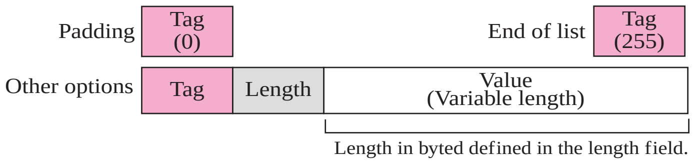


- **Client IP address.** This is a 4-byte field that contains the client IP address. If the client does not have this information, this field has a value of 0.
- **Your IP address.** This is a 4-byte field that contains the client IP address. It is filled by the server (in the reply message) at the request of the client.
- **Server IP address.** This is a 4-byte field containing the server IP address. It is filled by the server in a reply message.
- **Gateway IP address.** This is a 4-byte field containing the IP address of a router. It is filled by the server in a reply message.
- **Client hardware address.** This is the physical address of the client. Although the server can retrieve this address from the frame sent by the client, it is more efficient if the address is supplied explicitly by the client in the request message.
- **Server name.** This is a 64-byte field that is optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the domain name of the

server. If the server does not want to fill this field with data, the server must fill it with all 0s.

- **Boot filename.** This is a 128-byte field that can be optionally filled by the server in a reply packet. It contains a null-terminated string consisting of the full pathname of the boot file. The client can use this path to retrieve other booting information. If the server does not want to fill this field with data, the server must fill it with all 0s.
- **Options.** This is a 64-byte field with a dual purpose. It can carry either additional information (such as the network mask or default router address) or some specific vendor information. The field is used only in a reply message. The server uses a number, called a **magic cookie,** in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field. The length field defines the length of the value field, not the whole option. See Figure 18.6.

Figure 18.6 *Option format*



The list of options is shown in Table 18.1.

Table 18.1 *Options for DHCP*

Tag	*Length*	*Value*	*Description*
0			Padding
1	4	Subnet mask	Subnet mask
2	4	Time of the day	Time offset
3	Variable	IP addresses	Default router
4	Variable	IP addresses	Time server
5	Variable	IP addresses	IEN 16 server
6	Variable	IP addresses	DNS server
7	Variable	IP addresses	Log server
8	Variable	IP addresses	Quote server
9	Variable	IP addresses	Print server
10	Variable	IP addresses	Impress
11	Variable	IP addresses	RLP server
12	Variable	DNS name	Host name
13	2	Integer	Boot file size
53	1	Discussed later	Used for dynamic configuration
128–254	Variable	Specific information	Vendor specific
255			End of list

The lengths of the fields that contain IP addresses are multiples of 4 bytes. The padding option, which is only 1 byte long, is used only for alignment. The end-of-list option, which is also only 1 byte long, indicates the end of the option field. Vendors can use option tags 128 to 254 to supply extra information in a reply message.

18.3 CONFIGURATION

The DHCP has been devised to provide static and dynamic address allocation.

Static Address Allocation

In this capacity, a DHCP server has a database that statically binds physical addresses to IP addresses. When working in this way, DHCP is backward compatible with the deprecated protocol BOOTP, which we discussed before.

Dynamic Address Allocation

DHCP has a second database with a pool of available IP addresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

When a DHCP client sends a request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned. On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.

The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network (for example, a subscriber to a service provider). DHCP provides temporary IP addresses for a limited period of time.

The addresses assigned from the pool are temporary addresses. The DHCP server issues a **lease** for a specific period of time. When the lease expires, the client must either stop using the IP address or renew the lease. The server has the choice to agree or disagree with the renewal. If the server disagrees, the client stops using the address.

Transition States

To provide dynamic address allocation, the DHCP client acts as a state machine that performs transitions from one state to another depending on the messages it receives or sends. The type of the message in this case is defined by the option with tag 53 that is included in the DHCP packet. In other words, instead of adding one extra field to the BOOTP protocol to define DHCP type, the designer decided to add an extra option for this purpose. Figure 18.7 shows the type option and the interpretation of its value to define the type of the DHCP packet.

Figure 18.8 shows the transition diagram with main states. The RFC and some implementations offer some more states that we leave the investigation as exercises.

Figure 18.7 *Option with tag 53*

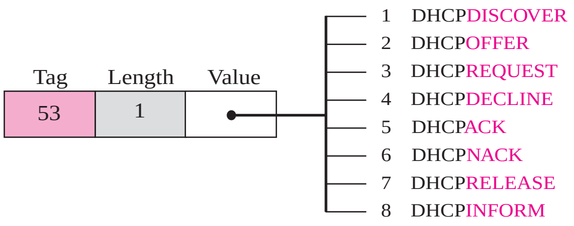
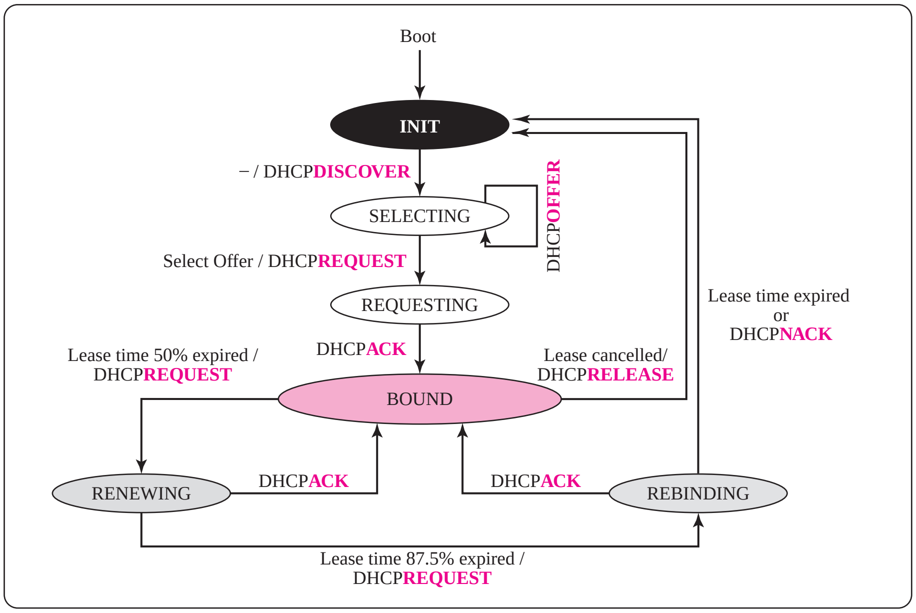


Figure 18.8 *DHCP client transition diagram*


Boot
INIT
− / DHCPDISCOVER
SELECTING
DHCPOFFER
Select Offer / DHCPREQUEST
REQUESTING
DHCPACK
Lease time 50% expired / DHCPREQUEST
Lease cancelled/ DHCPRELEASE
Lease time expired or DHCPNACK
BOUND
RENEWING
DHCPACK
DHCPACK
REBINDING
Lease time 87.5% expired / DHCPREQUEST

INIT State

When the DHCP client first starts, it is in the INIT state (initializing state). The client broadcasts a DHCPDISCOVER message (a request message with the DHCPDISCOVER option), using port 67.

SELECTING State

After sending the DHCPDISCOVER message, the client goes to the selecting state. Those servers that can provide this type of service respond with a DHCPOFFER message. In these messages, the servers offer an IP address. They can also offer the lease duration. The default is 1 hour. The server that sends a DHCPOFFER locks the offered IP address so that it is not available to any other clients. The client chooses one of the offers and sends a DHCPREQUEST message to the selected server. It then goes to the requesting state. However, if the client receives no DHCPOFFER message, it tries four more times, each with a span of 2 seconds. If there is no reply to any of these DHCPDISCOVERs, the client sleeps for 5 minutes before trying again.

REQUESTING State

The client remains in the requesting state until it receives a DHCPACK message from the server that creates the binding between the client physical address and its IP address. After receipt of the DHCPACK, the client goes to the bound state.

BOUND State

In this state, the client can use the IP address until the lease expires. When 50 percent of the lease period is reached, the client sends another DHCPREQUEST to ask for renewal. It then goes to the renewing state. When in the bound state, the client can also cancel the lease and go to the initializing state.

RENEWING State

The client remains in the renewing state until one of two events happens. It can receive a DHCPACK, which renews the lease agreement. In this case, the client resets its timer and goes back to the bound state. Or, if a DHCPACK is not received, and 87.5 percent of the lease time expires, the client goes to the rebinding state.

REBINDING State

The client remains in the rebinding state until one of three events happens. If the client receives a DHCPNACK or the lease expires, it goes back to the initializing state and tries to get another IP address. If the client receives a DHCPACK, it goes to the bound state and resets the timer.

Other Issues

In this section we discuss a few issues related to the DHCP states.

Early Release

A DHCP client that has been assigned an address for a period of time may release the address before the expiration time. The client may send a DHCPRELEASE message to tell the server that the address is no longer needed. This helps the server to assign the address to another client waiting for it.

Timers

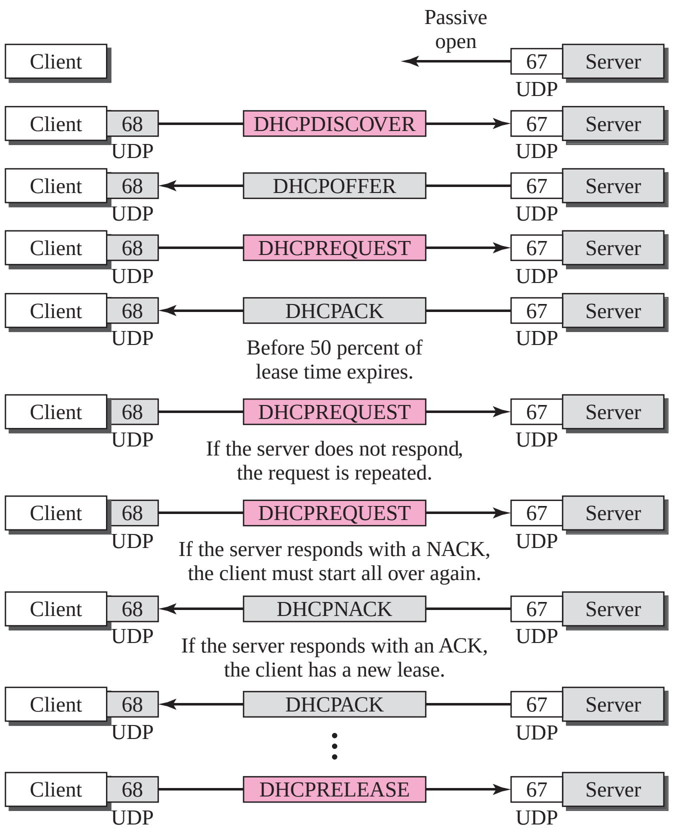
The above discussion requires that the client uses three times: renewal timer, rebinding timer, and expiration timer. If the server does not specify the time-out values for these timers when the address is allocated, the client needs to use the default value. The default value for each timer is shown below:

Renewal timer:	→	**50% of lease time**
Rebinding timer:	→	**87.5% of lease time**
Expiration timer:	→	**100% of lease time**

Exchanging Messages

Figure 18.9 shows the exchange of messages related to the transition diagram.

Figure 18.9 *Exchanging messages*


Passive open
Client
67
Server
UDP
Client
68
UDP
DHCPDISCOVER
DHCPOFFER
DHCPREQUEST
DHCPACK
Before 50 percent of lease time expires.
DHCPREQUEST
If the server does not respond, the request is repeated.
DHCPREQUEST
If the server responds with a NACK, the client must start all over again.
DHCPNACK
If the server responds with an ACK, the client has a new lease.
DHCPACK
DHCPRELEASE

18.4 FURTHER READING

For more details about subjects discussed in this chapter, we recommend the following books and RFCs. The items enclosed in brackets refer to the reference list at the end of the book.

Books and RFCs

Several books and RFCs give an easy but thorough coverage of DHCP including [Com 06], RFC 3396, and RFC 3342.