# [UEFI Basics] EDK Network Framework (DHCP4)

jiangwei0512 Posted on 2024-01-21 08:13:11 Read 1.5k Collection 26 Likes 23



Category Column: UEFI Development Basics Article Tags: network uefi



摘要 This article focuses on DHCP4, which is an application layer protocol carried on UDP, with port numbers 67 and 68. Its main function is to automatically set and uniformly manage IP addresses. It also summarizes the DHCP4 code, analyzes functions such as Dhcp4DriverBindingSupported and Dhcp4DriverBindingStart, as well as the DHCP_SERVICE structure and important parameters.

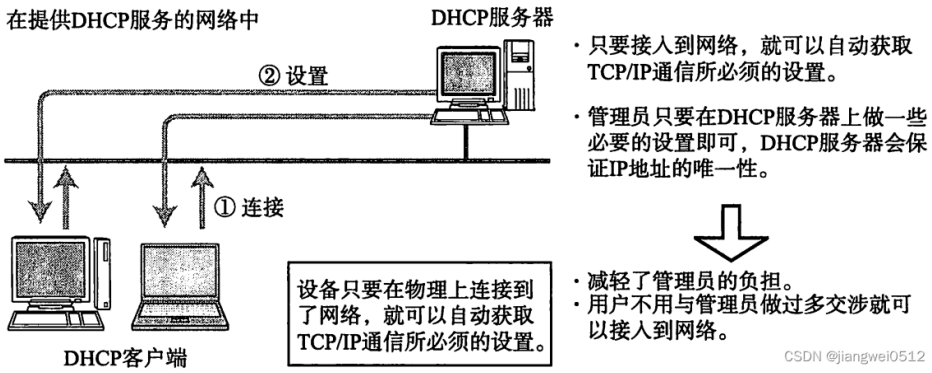
The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

## DHCP4

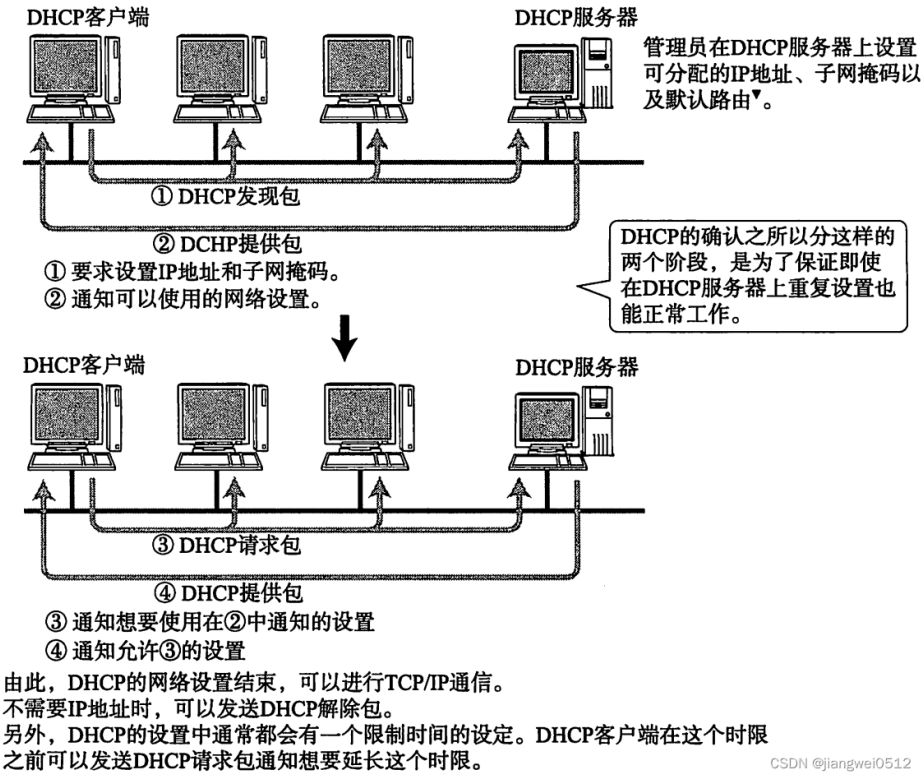
### DHCP4 Protocol Description

DHCP is an application layer protocol. DHCP messages are high-level protocol messages carried on UDP, using port numbers 67 ( DHCP server ) and 68 (DHCP client).

The full name of DHCP is **Dynamic Host Configuration Protocol** . Its main function is to automatically set IP addresses and uniformly manage IP address distribution .




How it works:




The DHCP message format is as follows:


The description of each parameter is as follows:

| Fields | Length (bytes) | meaning |
|---|---|---|
| Op | 1 | Indicates the type of message:<br>1: Client request message<br>2: Server response message |
| Htype | 1 | Indicates the type of the hardware address. For Ethernet, the value of this type is "1". |
| Hlen | 1 | Indicates the length of the hardware address in bytes. For Ethernet, this value is 6. |
| Hops | 1 | Number of hops. Set to 0 by the client, can also be set by a proxy server. |
| Xid | 4 | Transaction ID, a random number chosen by the client, is used by the server and client to communicate requests and responses between them, and the client uses it to match requests and responses.<br>This ID is set by the client and returned by the server as a 32-bit integer. |
| Secs | 2 | Filled by the client, indicating the number of seconds that have passed since the client obtained the IP address or renewed the IP address. |
| Flags | 2 | This field is reserved for BOOTP and represents the flag field in DHCP.<br>The Flags field format is:<br>0 15<br>+--------------------------------+<br>` |
| Ciaddr | 4 | The client's IP address. This field can be filled in only when the client is in the Bound, Renew, or Rebinding state and can respond to ARP requests. |
| Yiaddr | 4 | "Your own" or client's IP address. |
| Siaddr | 4 | Indicates the IP address of the server to be used in the next stage of the DHCP protocol process. |
| Giaddr | 4 | This field indicates the IP address of the first DHCP relay (note: not the gateway defined in the address pool).<br>When the client sends a DHCP request, if the server and the client are not in the same network, the first DHCP relay will fill in its own IP address in this field when forwarding the DHCP request message.<br>The server will determine the network segment address based on this field, and then select the address pool to allocate addresses to users.<br>The server will also send a response message to this DHCP relay based on this address, and the DHCP relay will forward this message to the client.<br>If more than one DHCP relay passes before reaching the DHCP server, the relay after the first DHCP relay will not change this field, but will increase the number of Hops by 1. |
| Chaddr | 16 | This field indicates the MAC address of the client. This field is consistent with the previous "Hardware Type" and "Hardware Length". When the client sends a DHCP request, it fills in its own hardware address in this field. For Ethernet, when "Hardware Type" and "Hardware Length" are "1" and "6" respectively, this field must be filled in with a 6-byte Ethernet MAC address. |
| Sname | 64 | This field indicates the server name from which the client obtains configuration information. This field is filled in by the DHCP server and is optional.<br>If filled in, it must be a string ending with 0. |
| File | 128 | This field indicates the name of the client's startup configuration file. This field is filled in by the DHCP server and is optional.<br>If filled in, it must be a string ending with 0. |
| Options | variable | This field indicates the DHCP option field, which is at least 312 bytes in the format of "code + length + data".<br>DHCP uses this field to include the configuration information assigned by the server to the terminal, such as the gateway IP address, the IP address of the DNS server, and the effective lease period of the IP address that the client can use. |

Corresponding to the structure in the code:

c AI generated projects 登录复制 run

```c
#pragma pack(1)
///
/// EFI_DHCP4_PACKET defines the format of DHCPv4 packets. See RFC 2131 for more information.
///
typedef struct {
  UINT8              OpCode;
  UINT8              HwType;
  UINT8              HwAddrLen;
  UINT8              Hops;
  UINT32             Xid;
  UINT16             Seconds;
  UINT16             Reserved;
  EFI_IPv4_ADDRESS   ClientAddr;       ///< Client IP address from client.
  EFI_IPv4_ADDRESS   YourAddr;         ///< Client IP address from server.
  EFI_IPv4_ADDRESS   ServerAddr;       ///< IP address of next server in bootstrap.
  EFI_IPv4_ADDRESS   GatewayAddr;      ///< Relay agent IP address.
  UINT8              ClientHwAddr[16]; ///< Client hardware address.
  CHAR8              ServerName[64];
  CHAR8              BootFileName[128];
} EFI_DHCP4_HEADER;
twen #pragma pack()
```

### DHCP4 Code Overview

DHCP4 is also a common network protocol. In fact, it is now NetworkPkg\Dhcp4Dxe\Dhcp4Dxe.inf. Here we first need to look at its entry:

c AI generated projects 登录复制 run

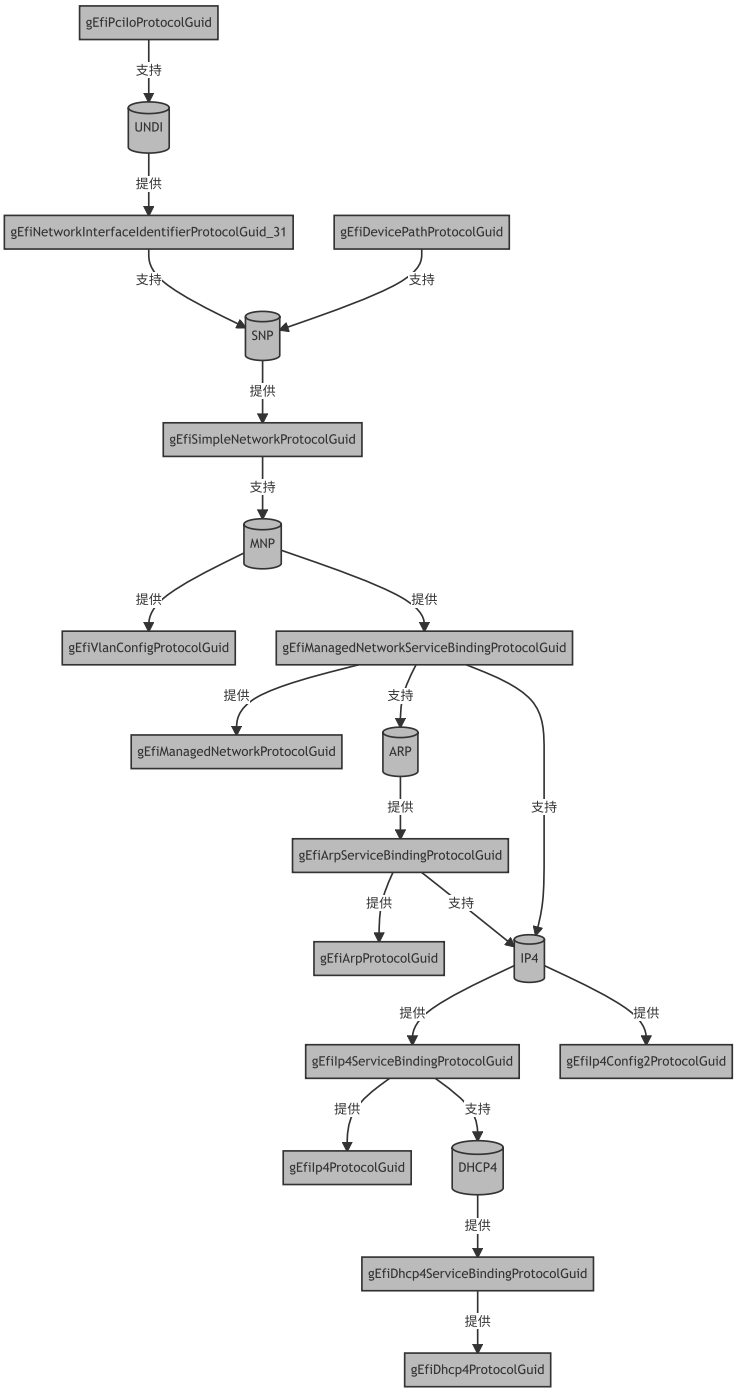
```c
EFI_STATUS
EFIAPI
Dhcp4DriverEntryPoint (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  return EfiLibInstallDriverBindingComponentName2 (
           ImageHandle,
           SystemTable,
           &gDhcp4DriverBinding,
           ImageHandle,
           &gDhcp4ComponentName,
           &gDhcp4ComponentName2
           );
}
```

Just installed `gDhcp4DriverBinding` :

c AI generated projects 登录复制 run

```c
EFI_DRIVER_BINDING_PROTOCOL  gDhcp4DriverBinding = {
  Dhcp4DriverBindingSupported,
  Dhcp4DriverBindingStart,
  Dhcp4DriverBindingStop,
  0xa,
  NULL,
  NULL
};
```

Diagram of DHCP4 in the UEFI network protocol stack:




Its structure is basically the same as that of DNS4 .

### Dhcp4DriverBindingSupported

DHCP4 relies on UDP4 :

```c
EFI_STATUS
EFIAPI
Dhcp4DriverBindingSupported (
  IN EFI_DRIVER_BINDING_PROTOCOL  *This,
  IN EFI_HANDLE                   ControllerHandle,
  IN EFI_DEVICE_PATH_PROTOCOL     *RemainingDevicePath OPTIONAL
  )
{
  Status = gBS->OpenProtocol (
                  ControllerHandle,
                  &gEfiUdp4ServiceBindingProtocolGuid,
                  NULL,
                  This->DriverBindingHandle,
                  ControllerHandle,
                  EFI_OPEN_PROTOCOL_TEST_PROTOCOL
                  );
}
```

### Dhcp4DriverBindingStart

The flow of the Start function is as follows:

1. Creation `DHCP_SERVICE` will be further introduced later.
2. To receive data through UDP4 , the corresponding function is `UdpIoRecvDatagram()` , which will be executed in a loop:

```c
EFI_STATUS
EFIAPI
UdpIoRecvDatagram (
  IN  UDP_IO           *UdpIo,
  IN  UDP_IO_CALLBACK  CallBack,
  IN  VOID             *Context,
  IN  UINT32           HeadLen
  )
{
  RxToken = UdpIoCreateRxToken (UdpIo, CallBack, Context, HeadLen);

  UdpIo->RecvRequest = RxToken;
  if (UdpIo->UdpVersion == UDP_IO_UDP4_VERSION) {
    Status = UdpIo->Protocol.Udp4->Receive (UdpIo->Protocol.Udp4, &RxToken->Token.Udp4);
  }
}
```

Because there is the following code in the Start function:

```c
  //
  // Start the receiving
  //
  Status = UdpIoRecvDatagram (DhcpSb->UdpIo, DhcpInput, DhcpSb, 0);
```

So `UdpIoRecvDatagram()` the callback function corresponding to the Token created in is `DhcpInput()` , and the implementation of this function includes:

```c
VOID
EFIAPI
DhcpInput (
  NET_BUF        *UdpPacket,
  UDP_END_POINT  *EndPoint,
  EFI_STATUS     IoStatus,
  VOID           *Context
  )
{
  // 前面是数据处理，处理完之后就跳转到RESTRT，或者就走这里的异常，但是也是调用了UdpIoRecvDatagram()
  if (EFI_ERROR (Status)) {
    NetbufFree (UdpPacket);
    UdpIoRecvDatagram (DhcpSb->UdpIo, DhcpInput, DhcpSb, 0);
    DhcpEndSession (DhcpSb, Status);
    return;
  }

RESTART:
  Status = UdpIoRecvDatagram (DhcpSb->UdpIo, DhcpInput, DhcpSb, 0);
}
```

That's why there is the statement "Start the receiving" in the notes.

3. Install `gEfiDhcp4ServiceBindingProtocolGuid` .

### DHCP_SERVICE

`DHCP_SERVICE` Create in the Start function:

```c
EFI_STATUS
EFIAPI
Dhcp4DriverBindingStart (
  IN EFI_DRIVER_BINDING_PROTOCOL  *This,
  IN EFI_HANDLE                   ControllerHandle,
  IN EFI_DEVICE_PATH_PROTOCOL     *RemainingDevicePath OPTIONAL
  )
{
  Status = Dhcp4CreateService (ControllerHandle, This->DriverBindingHandle, &DhcpSb);
}
```

Its structure is located in NetworkPkg\Dhcp4Dxe\Dhcp4Impl.h:

```c
//
// DHCP driver is specical in that it is a singleton. Although it
// has a service binding, there can be only one active child.
//
struct _DHCP_SERVICE {
  UINT32                          Signature;
  EFI_SERVICE_BINDING_PROTOCOL    ServiceBinding;

  INTN                            ServiceState; // CONFIGED, UNCONFIGED, and DESTROY
```

```
  EFI_HANDLE                    Controller;
  EFI_HANDLE                    Image;

  LIST_ENTRY                    Children;
  UINTN                         NumChildren;

  INTN                          DhcpState;
  EFI_STATUS                    IoStatus;   // the result of last user operation
  UINT32                        Xid;

  IP4_ADDR                      ClientAddr; // lease IP or configured client address
  IP4_ADDR                      Netmask;
  IP4_ADDR                      ServerAddr;

  EFI_DHCP4_PACKET              *LastOffer; // The last received offer
  EFI_DHCP4_PACKET              *Selected;
  DHCP_PARAMETER                *Para;

  UINT32                        Lease;
  UINT32                        T1;
  UINT32                        T2;
  INTN                          ExtraRefresh; // This refresh is reqested by user

  UDP_IO                        *UdpIo;       // Udp child receiving all DHCP message
  UDP_IO                        *LeaseIoPort; // Udp child with lease IP
  EFI_DHCP4_PACKET              *LastPacket;  // The last sent packet for retransmission
  EFI_MAC_ADDRESS               Mac;
  UINT8                         HwType;
  UINT8                         HwLen;
  UINT8                         ClientAddressSendOut[16];

  DHCP_PROTOCOL                 *ActiveChild;
  EFI_DHCP4_CONFIG_DATA         ActiveConfig;
  UINT32                        UserOptionLen;

  //
  // Timer event and various timer
  //
  EFI_EVENT                     Timer;

  UINT32                        PacketToLive; // Retransmission timer for our packets
  UINT32                        LastTimeout;  // Record the init value of PacketToLive every time
  INTN                          CurRetry;
  INTN                          MaxRetries;
  UINT32                        LeaseLife;
};
```

First of all, we need to pay attention to the comments at the beginning. Although there can be multiple such service data, DHCP4 is just a singleton, which means there is only one valid one.

Important parameters are described as follows:

- `ServiceState` : Corresponding value:

c — AI generated projects — 登录复制 — run

```c
//
// The state of the DHCP service. It starts as UNCONFIGED. If
// and active child configures the service successfully, it
// goes to CONFIGED. If the active child configures NULL, it
// goes back to UNCONFIGED. It becomes DESTROY if it is (partly)
// destroyed.
//
#define DHCP_UNCONFIGED  0
#define DHCP_CONFIGED    1
#define DHCP_DESTROY     2
```

Its usage has been explained in the above comments.

- `DhcpState` : It indicates the DHCP status during the communication process, the corresponding value is:

c — AI generated projects — 登录复制 — run

```c
typedef enum {
  ///
  /// The EFI DHCPv4 Protocol driver is stopped.
  ///
  Dhcp4Stopped = 0x0,
  ///
  /// The EFI DHCPv4 Protocol driver is inactive.
  ///
  Dhcp4Init = 0x1,
  ///
  /// The EFI DHCPv4 Protocol driver is collecting DHCP offer packets from DHCP servers.
  ///
  Dhcp4Selecting = 0x2,
  ///
  /// The EFI DHCPv4 Protocol driver has sent the request to the DHCP server and is waiting for a response.
  ///
  Dhcp4Requesting = 0x3,
  ///
  /// The DHCP configuration has completed.
  ///
  Dhcp4Bound = 0x4,
  ///
  /// The DHCP configuration is being renewed and another request has
  /// been sent out, but it has not received a response from the server yet.
  ///
  Dhcp4Renewing = 0x5,
  ///
  /// The DHCP configuration has timed out and the EFI DHCPv4
  /// Protocol driver is trying to extend the lease time.
  ///
  Dhcp4Rebinding = 0x6,
  ///
  /// The EFI DHCPv4 Protocol driver was initialized with a previously
  /// allocated or known IP address.
  ///
  Dhcp4InitReboot = 0x7,
  ///
  /// The EFI DHCPv4 Protocol driver is seeking to reuse the previously
  /// allocated IP address by sending a request to the DHCP server.
  ///

```

```
  Dhcp4Rebooting = 0x8
} EFI_DHCP4_STATE;
```

- `IoStatus` : Indicates the status caused by DHCP communication operation.
- `Xid` : A random number, which has been introduced in the DHCP4 protocol description .
- `ClientAddr` , `Netmask` , `ServerAddr` : `ClientAddr` are IP addresses leased through DHCP, corresponding codes:

c AI generated projects 登录复制 run

```c
EFI_STATUS
DhcpLeaseAcquired (
  IN OUT DHCP_SERVICE  *DhcpSb
  )
{
  DhcpSb->ClientAddr = EFI_NTOHL (DhcpSb->Selected->Dhcp4.Header.YourAddr);

  if (DhcpSb->Para != NULL) {
    DhcpSb->Netmask    = DhcpSb->Para->NetMask;
    DhcpSb->ServerAddr = DhcpSb->Para->ServerId;
  }
}
```

The other two IPs are also set here.

- `LastOffer` : The last received packet.
- `Selected` : The packet from which the IP is obtained.
- `Para` : DHCP parameters, its structure is as follows:

AI generated projects 登录复制

```
///
/// The options that matters to DHCP driver itself. The user of
/// DHCP clients may be interested in other options, such as
/// classless route, who can parse the DHCP offer to get them.
///
typedef struct {
  IP4_ADDR    NetMask;                // DHCP4_TAG_NETMASK
  IP4_ADDR    Router;                 // DHCP4_TAG_ROUTER, only the first router is used

  //
  // DHCP specific options
  //
  UINT8       DhcpType;               // DHCP4_TAG_MSG_TYPE
  UINT8       Overload;               // DHCP4_TAG_OVERLOAD
  IP4_ADDR    ServerId;               // DHCP4_TAG_SERVER_ID
  UINT32      Lease;                  // DHCP4_TAG_LEASE
  UINT32      T1;                     // DHCP4_TAG_T1
  UINT32      T2;                     // DHCP4_TAG_T2
} DHCP_PARAMETER;
```

`Lease` Parameters such as , `T1` , `T2` and are also reflected here.

- `ExtraRefresh` : It will be used when updating the lease period, corresponding to the function:

c AI generated projects 登录复制 run

```c
/**
  Extends the lease time by sending a request packet.

  The RenewRebind() function is used to manually extend the lease time when the
  EFI DHCPv4 Protocol driver is in the Dhcp4Bound state and the lease time has
  not expired yet. This function will send a request packet to the previously
  found server (or to any server when RebindRequest is TRUE) and transfer the
  state into the Dhcp4Renewing state (or Dhcp4Rebinding when RebindingRequest is
  TRUE). When a response is received, the state is returned to Dhcp4Bound.
  If no response is received before the try count is exceeded (the RequestTryCount
  field that is specified in EFI_DHCP4_CONFIG_DATA) but before the lease time that
  was issued by the previous server expires, the driver will return to the Dhcp4Bound
  state and the previous configuration is restored. The outgoing and incoming packets
  can be captured by the EFI_DHCP4_CALLBACK function.

  @param[in]  This            Pointer to the EFI_DHCP4_PROTOCOL instance.
  @param[in]  RebindRequest   If TRUE, this function broadcasts the request packets and enters
                              the Dhcp4Rebinding state. Otherwise, it sends a unicast
                              request packet and enters the Dhcp4Renewing state.
  @param[in]  CompletionEvent If not NULL, this event is signaled when the renew/rebind phase
twen                          completes or some error occurs.
twen                          EFI_DHCP4_PROTOCOL.GetModeData() can be called to
twen                          check the completion status. If NULL,
twen                          EFI_DHCP4_PROTOCOL.RenewRebind() will busy-wait
                              until the DHCP process finishes.

  @retval EFI_SUCCESS           The EFI DHCPv4 Protocol driver is now in the
                                Dhcp4Renewing state or is back to the Dhcp4Bound state.
  @retval EFI_NOT_STARTED       The EFI DHCPv4 Protocol driver is in the Dhcp4Stopped
                                state. EFI_DHCP4_PROTOCOL.Configure() needs to
                                be called.
  @retval EFI_INVALID_PARAMETER This is NULL.
  @retval EFI_TIMEOUT           There was no response from the server when the try count was
                                exceeded.
  @retval EFI_ACCESS_DENIED     The driver is not in the Dhcp4Bound state.
  @retval EFI_DEVICE_ERROR      An unexpected system or network error occurred.

**/
EFI_STATUS
EFIAPI
EfiDhcp4RenewRebind (
  IN EFI_DHCP4_PROTOCOL  *This,
  IN BOOLEAN             RebindRequest,
  IN EFI_EVENT           CompletionEvent   OPTIONAL
  )
{
  DhcpSb->ExtraRefresh       = TRUE;
}
```

- `UdpIo` : UDP4 communication interface, which is wrapped here. Because there are two versions, v4 and v6, we only focus on the v4 version.
- `LeaseIoPort` : It is also a UDP4 communication interface, but it corresponds to the interface for obtaining a leased IP.
- `LastPacket` : The last packet sent.
- `Mac` , `HwType` , `HwLen` : Network card parameters, corresponding to `EFI_SIMPLE_NETWORK_MODE` the values in .

- `ClientAddressSendOut` : Corresponding to the DHCP packet header `ClientHwAddr` .
- `ActiveChild` , `ActiveConfig` : As mentioned earlier, DHCP uses a singleton, so `DHCP_PROTOCOL` there is only one valid one, which executes the valid one `DHCP_PROTOCOL` and the corresponding parameters.
- `UserOptionLen` : The length of the DHCP option.
- `Timer` : A timer is created when the service is created:

c AI generated projects 登录复制 run

```c
  //
  // Create various resources, UdpIo, Timer, and get Mac address
  //
  Status = gBS->CreateEvent (
                  EVT_NOTIFY_SIGNAL | EVT_TIMER,
                  TPL_CALLBACK,
                  DhcpOnTimerTick,
                  DhcpSb,
                  &DhcpSb->Timer
                  );
```

You can know its function from the corresponding callback function:

c AI generated projects 登录复制 run

```c
/**
  Each DHCP service has three timer. Two of them are count down timer.
  One for the packet retransmission. The other is to collect the offers.
  The third timer increments the lease life which is compared to T1, T2,
  and lease to determine the time to renew and rebind the lease.
  DhcpOnTimerTick will be called once every second.

  @param[in]  Event                The timer event
  @param[in]  Context              The context, which is the DHCP service instance.

**/
VOID
EFIAPI
DhcpOnTimerTick (
  IN EFI_EVENT  Event,
  IN VOID       *Context
  )
```

It executes once a second:

c AI generated projects 登录复制 run

```c
  Status = gBS->SetTimer (DhcpSb->Timer, TimerPeriodic, TICKS_PER_SECOND);
```

- `PacketToLive` , `LastTimeout` , `CurRetry` , `MaxRetries` : Some values related to timers, and also related to mechanisms such as retransmission.
- `LeaseLife` : Lease time.