Embedding it better...

µTasker Document

µTasker – DHCP Client

Table of Contents

1. Introduction

This document describes the DHCP protocol and its implementation in the µTasker project. In most cases this operation takes place in the background and requires very little interaction with the user application.

In systems with multiple interfaces and networks DHCP client operation can be enabled on individual networks and restricted to certain interfaces where required, allowing multiple networks to each obtain their own IP configuration.

DHCP client support is enabled in the µTasker project with the define `USE_DHCP_CLIENT` (early implementations may use `USE_DHCP`).

2. DHCP Protocol

DHCP (Dynamic Host Configuration Protocol – RFC 2131) is used for automatic configuration of network clients in which case one or more DHCP servers are configured to supply the correct parameters for a given network.

The clients first locate the server they want to use and then request the necessary parameters, either suggesting preferred ones or letting the DHCP server offer them.

The RFC2131 state-event diagram is reproduced in figure 1.

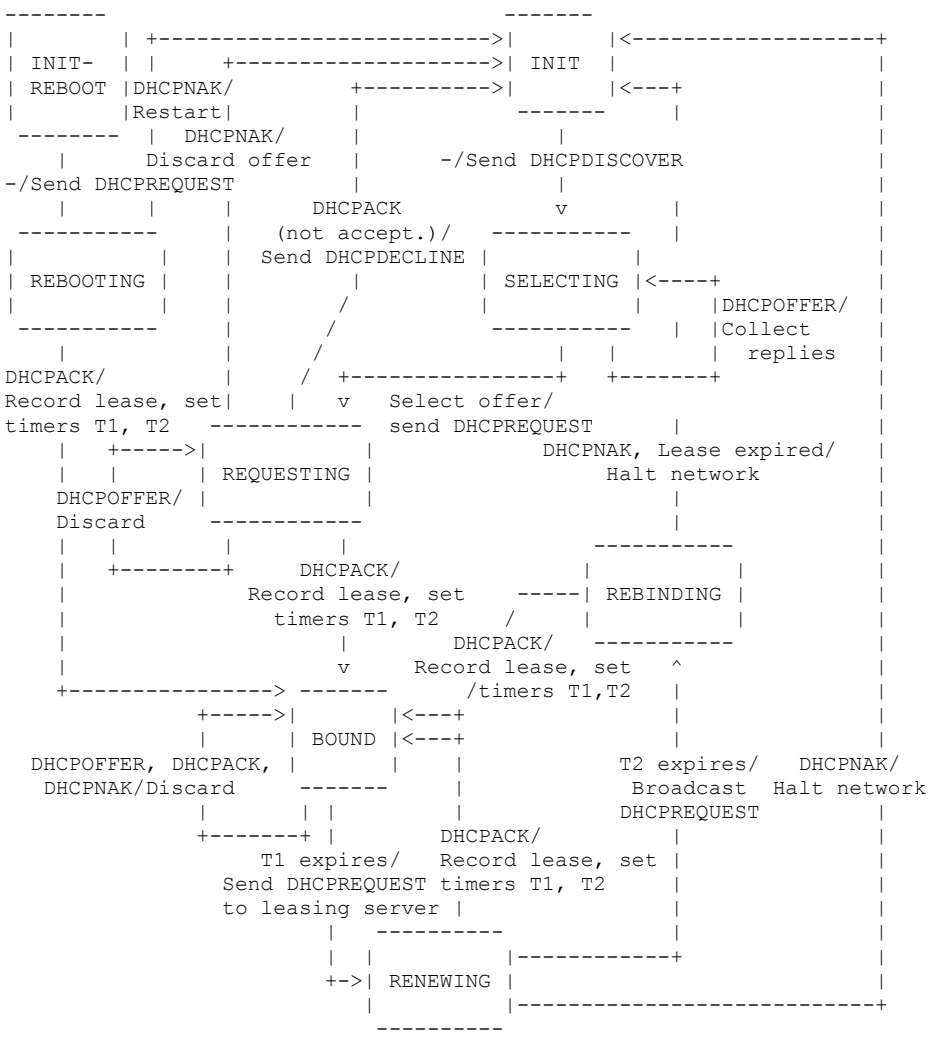

Figure 1: State-transition diagram for DHCP clients [RFC 2131]

2.1. UDP Frame Size Requirement

It is to be noted that the use of DHCP imposes a constraint on the minimum Ethernet buffer size in order to be able to receive all possible DHCP messages, which can have a length of up to 576 bytes (longer message size can be negotiated): 44 bytes header, 64 bytes for server host name, 128 bytes boot file name and 312 bytes vendor specific information. In small foot print systems the length of the Ethernet buffers can often be programmed to be as

small as 128 bytes and up to 1,5k, adequate to receive the maximum Ethernet frame of 1514 bytes. TCP, for example, supports capabilities of operating with small buffer sizes without restrictions – *apart from potential throughput issues* – allowing suitable compromises in such systems to enable trade-offs between minimum RAM resources and performance. DHCP doesn't support a suitable mechanism and so the buffer size must be set adequately.

2.2. Starting DHCP Negotiation

The DHCP client discovery operation is started by an application task by calling the function:

```
extern int fnStartDHCP(UTASK_TASK OwnerTask, USOCKET uDetails);
```

*Note that uDetails is not present in early implementations.

The application task name is passed so that DHCP events can be sent to this task. Examples of such events are `DHCP_SUCCESSFUL` [*DHCP negotiation has been successfully terminated and network operation can begin/continue*] and `DHCP_LEASE_TERMINATED` [*DHCP lease has expired without being able to renegotiate lease extension – all network operation should terminate*].

The `uDetails` parameter informs of the type of DHCP operation required. In a simple configuration with a single Ethernet interface the value (`DHCP_CLIENT_OPERATION | defineInterface(DEFAULT_IP_INTERFACE) | defineNetwork(DEFAULT_NETWORK))` is used to define client mode operation on the default interface and default network.

In systems with a single network on multiple interfaces each interface that may be involved with the DHCP activity is passed, which allow the resolution to be restricted to certain ones if needed.

In systems with multiple networks the DHCP client operation needs to be started for each network that requires resolution of its IP configuration. This allows certain networks to use fixed IP settings and others to resolve theirs. Each network has its own DHCP client instance so that each network's resolution operation is independent. Each network defines the interfaces that are involved in its own DHCP resolution.

3. DHCP Message Timeouts

DHCP messages are initially broadcast in nature and after a DHCP server has leased a set of parameters they become uni-cast with the specific DHCP server. Requests and responses are matched using a transaction ID (xid) which is a random 32 bit number generated by the client and transmissions are re-tried after defined delays should a response not be received.

As specified in RFC 2131, the DHCP client is responsible for all retransmissions and uses an initial 4s repetition time with +/-1s randomisation before switching to an exponential back-off up to a maximum of 64s.

The µTasker DHCP client will send its initial discovery message with the following periodicity when there is no DHCP server responding to it:

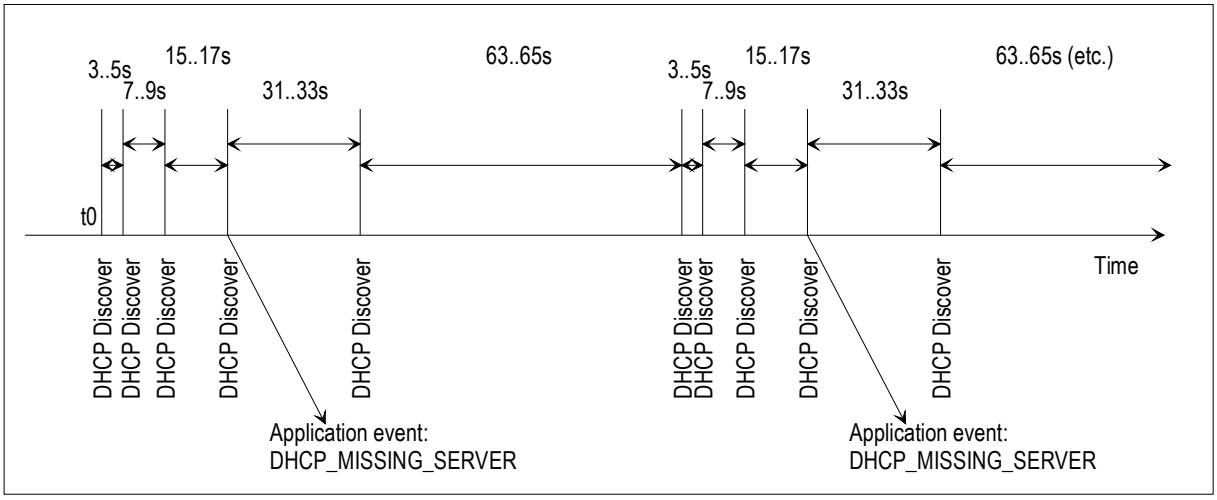

Figure 2 - Discovery Message Repetition Timing

The task which started the DHCP operation receives an event each time that the repetition time increases to 32s (+/- 1s) and can decide to disable DHCP and switch to a default set of parameters since it is probable that no DHCP server exists in the network to perform negotiation with. In this case the DHCP operation can be stopped by calling the function:

```
extern void fnStopDHCP(int iNetwork);
```

*Note that iNetwork is not present in early implementations.

Further discovery attempts can be restarted at any time using the start call again. Note that the local IP address is set on calling the stop command to the value contained in the preferred DHCP location. This allows the system to continue using its preferred address as default if no DHCP server was found. During DHCP negotiation the local IP address is set to 0.0.0.0 to indicate that network operation is not yet permitted.

Example of stopping DHCP operation on the default network:

```
fnStop(DEFAULT_NETWORK);
```

4. DHCP Implementation

The DHCP protocol is realised as a state event machine by the µTasker, as described in RFC 2131. *There is an independent state-event machine for each network, when multiple networks are in operation.*

The BOOTP broadcast flag is always set and so all messages from the DHCP server during negotiation will be broadcast types (and not uni-cast).

As soon as a DHCP server makes an offer in response to a discovery message from the DHCP client, the client will request lease parameters.

Lease time-outs are controlled by three timeout values returned by the DHCP server during negotiation as follows:

- *Lease Time* – the total lease time which has been given (0xffffffff is infinite)
- *Renewal Time* – the time after which the renewal process should be started (generally half of the least time). The DHCP client should request an extension of the lease from the DHCP server which has given the lease. This is referred to as time **T1**.
- *Rebinding Time* – the time after which new parameters should be requested from any DHCP server (generally at 0.875 times the lease duration. This is referred to as time **T2** and is only used when the lease could not be extended after **T1**.

If a lease times out because it could neither be extended nor a new lease be negotiated, the DHCP client no longer has the right to use the leased parameters and is no longer network capable.

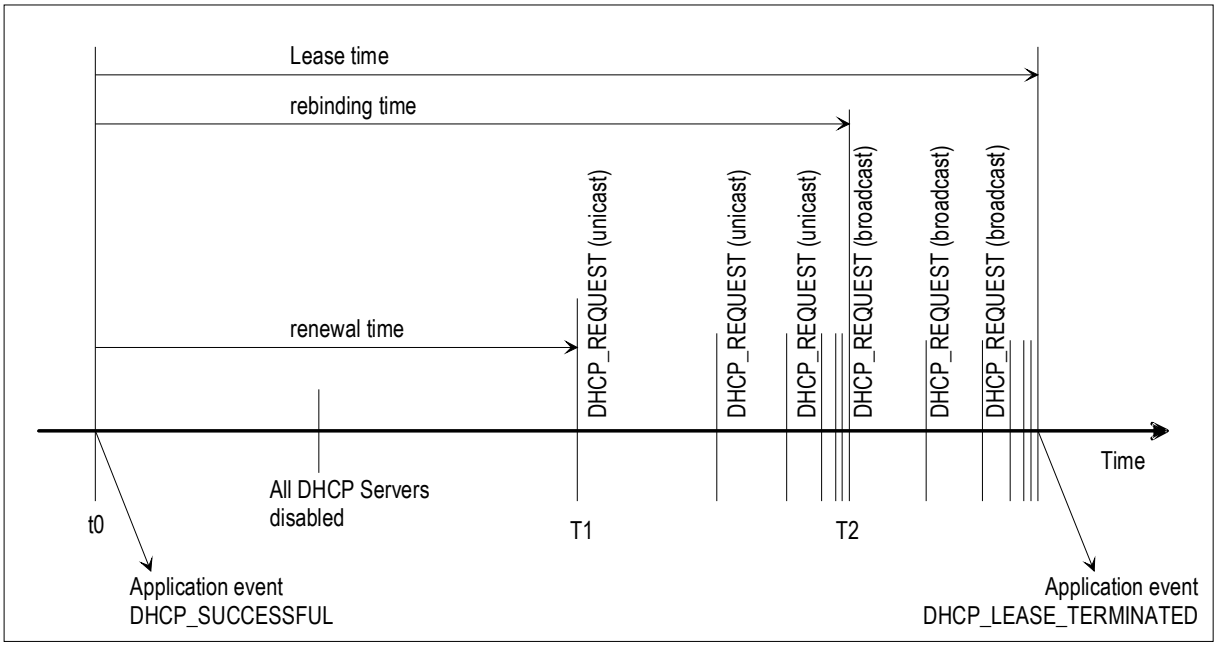

Figure 3 - Lease re-negotiation times

Figure 3 illustrates the operation of the renewal/rebinding process when the leasing server and all other DHCP servers on the network are deactivated after the original lease parameters have been obtained.

Up to the time **T1** there is no activity and the DHCP client may use the leased parameters. At **T1** the DHCP client tries to renew the lease. In the normal case the DHCP server would still exist and know the DHCP client and renew the release which would effectively restart at t0 (although the application is not informed after a renewal). However, since the DHCP server which leased the parameters is no longer present, there is no answer. The DHCP client retries again after the remaining time to **T2** has halved and thereafter with increasing frequency (halving remaining time) until the remaining time has decreased to less than 60s, in which case no more repeats are made.

At **T2** the rebinding period starts where the DHCP request is no longer sent as uni-cast to the original leasing server but is sent as a broadcast to allow any DHCP server in the network to respond. The retransmission period is half the remaining time until the end of the lease and is halved until the retransmission timeout decreases to less than 60s, after which no further retransmissions are performed.

Once the lease time runs out without being able to rebind, the DHCP device is no longer allowed to use the leased parameters. The application is informed via a `DHCP_LEASE_TERMINATED` event and can either switch to a default set of parameters or stop all network activity.

4.1. MAC Address Verification of Offer Messages

Due to the fact that the original address offers are based on multicast UDP frames all DHCP clients presently in the process of DHCP discovery receive all offers. The client checks that the offer matches with the XID (reference number of the request) which is either a fixed value or one generated by whatever local random number technique is available. There is thus a risk that tow DHCP clients are using the same XID at the same time (especially when the XID is fixed or their random number generators are potentially synchronised due to starting at the same time). For this reason the checking of the MAC address of the destination, as contained within the offer, was added to the verification of the offer frame.

4.2. Plausibility Check of DHCP Server Times

The DHCP server defines the lease time of IP configuration settings. At the same time it defines the renewal time and the rebinding time. Often these times are 0.5 and 0.875 of the lease time respectively. These values are however parameters of the server and it is possible that these values are not realistic.

Examples of unrealistic values would be a renewal time greater than the least time or when all times are equal. For this reason a plausibility check of the renewal and rebinding times has been added. If these are either impossible or would give time differences of less than 10 minutes only the lease time is used and the rebinding time set to 0.75 of the lease time and the renewal time set to 2/3 of the rebinding time (typically half of the lease time).

5. Accelerated DHCP Discovery with Preferred IP Address

When the client initially has a non-zero IP address, know as a preferred IP address, the first stage of the discovery process consists of a request for use of this IP address. In many cases, when this IP address was not previously allocated by the DHCP server, the initial request will be refused and the second stage, a new initialisation, is started.

The initial request takes several seconds to be performed according to the protocol requirement and so constitutes an increased delay from initialisation of the Ethernet interface until the time that an operational IP configuration has been received.

In cases where a preferred IP address is required, for example as fall back address, but the initial stage of requesting this specific address is not desired, the second phase start (as is the case when the IP address is set to 0.0.0.0) can be forced by using the following DHCP start call:

```
    fnStartDHCP((UTASK_TASK)(FORCE_INIT | OWN_TASK),(DHCP_CLIENT_OPERATION
| defineInterface(DEFAULT_IP_INTERFACE) | defineNetwork(DEFAULT_NETWORK)));
```

6. Host Name Option

When the option DHCP_HOST_NAME is enabled the DHCP client will report the host name option in each of its requests. This can be useful if the DHCP server allows a display of its clients since it enabled simple recognition of the client device as a string to associate its IP address on the network.

When the option is enabled the application must supply the routine

```
extern CHAR *fnGetDHCP_host_name(unsigned char *ptr_ucHostNameLength,
                                 int iNetwork)
```

which the DHCP client calls each time it needs to add the host name to the request. An example of inserting a host name is given below, whereby different names could be returned for different networks if needed.

```
// Supply a DHCP host name
//
extern CHAR *fnGetDHCP_host_name(unsigned char *ptr_ucHostNameLength,int iNetwork)
{
    *ptr_ucHostNameLength = uStrlen(temp_pars->temp_parameters.cDeviceIDName);
                                                          // length of the name
    return (temp_pars->temp_parameters.cDeviceIDName);
                                             // return pointer to the name to use
}
```

If the host name option is not to be inserted the user routine can return a zero pointer. Should the user return an empty string or a string that can't fit into the available DHCP client's transmit buffer space the option insertion will be skipped.

7. Handling DHCP Events

The following events are generated by the DHCP client:

```
DHCP_SUCCESSFUL       // DHCP was successful - the system can use TCP/IP
DHCP_COLLISION        // DHCP received infomation but there is an IP conflict - retry?
DHCP_LEASE_TERMINATED // DHCP lease has terminated without being able to establish new
                         lease - presently no network capability
DHCP_MISSING_SERVER   // DHCP repetition timer has been increased to a level
                         indicating no server present
```

Early implementations of the DHCP client send these events as interrupt events to the task indicated when starting the DHCP client.

Present implementations send a message event from the Ethernet task (`TASK_ETHERNET`) containing the event plus the network number of the DHCP client involved. This allows handling events from multiple DHCP clients, whereby the `DHCP_SUCCESSFUL` event informs that the network in question has received its IP configuration. `DHCP_MISSING_SERVER` informs that there was no response from DHCP servers in the network, in which case the application may decide to stop the DHCP client operation and fall-back to a fixed address rather than letting the DHCP client retry.

DHCP_COLLISION means that another device on the network was found with the same address that has just been allocated, in which case the DHCP client will usually be allowed to retry.
DHCP_LEASE_TERMINATED means that no server responded when trying to renew a lease. Usually the application will allow the DHCP client to continue trying to obtain a new IP configuration, although during this period the IP configuration will not be valid and so should not be used.

8. Conclusion

The DHCP protocol and the µTasker implementation according to RFC2131 have been described. The message repetition and re-lease timing have been illustrated, as well as a method to accelerate the discovery process when a fall-back IP address is available but doesn't need to specifically be requested.

Multiple networks are supported and each network can be restricted to certain interfaces if required.

Modifications:

V0.04 4.4.2009:
- Reformat document with header and table of contents. Add state-transition diagram and conclusion.

V0.05 3.5.2010 Added MAC address checking in offer and verification of invalid server time parameters

V0.06 5.7.2016 Added multiple network/interface operation, host server option and DHCP event handling