Guide d'Implémentation d'un Client DHCP en C Embarqué

Stack Minimale et Suffisante

Vue d'ensemble

Ce document présente les éléments essentiels pour implémenter un client DHCP minimal fonctionnel en C embarqué, en utilisant votre stack UDP existante.

---

1. Structure du paquet DHCP (RFC 2131)

Le paquet DHCP utilise le format BOOTP. Voici la structure à implémenter :

```c
typedef struct {
    uint8_t op;           // 1 = BOOTREQUEST, 2 = BOOTREPLY
    uint8_t htype;        // Type de hardware (1 = Ethernet)
    uint8_t hlen;         // Longueur adresse MAC (6 pour Ethernet)
    uint8_t hops;         // 0 pour client
    uint32_t xid;         // ID de transaction (random)
    uint16_t secs;        // Secondes écoulées
    uint16_t flags;       // Flags (0x8000 = broadcast)
    uint32_t ciaddr;      // IP client (0 si pas d'IP)
    uint32_t yiaddr;      // "Your" IP (rempli par serveur)
    uint32_t siaddr;      // IP du serveur DHCP
    uint32_t giaddr;      // IP gateway relay
    uint8_t chaddr[16];   // Adresse MAC client
    uint8_t sname[64];    // Nom serveur (optionnel)
    uint8_t file[128];    // Fichier boot (optionnel)
    uint32_t magic;       // Cookie magique: 0x63825363
    uint8_t options[];    // Options DHCP (longueur variable)
} __attribute__((packed)) dhcp_packet_t;
```

---

2. Options DHCP essentielles

Les options suivent le format TLV (Type-Length-Value). Codes importants :

Code	Nom	Taille	Description
53	DHCP Message Type	1 octet	Obligatoire : 1=DISCOVER, 2=OFFER, 3=REQUEST, 5=ACK, 6=NAK
50	Requested IP Address	4 octets	IP demandée par le client
54	Server Identifier	4 octets	IP du serveur DHCP
51	IP Address Lease Time	4 octets	Durée du bail en secondes
1	Subnet Mask	4 octets	Masque de sous-réseau
3	Router/Gateway	4+ octets	Adresse(s) de la passerelle
6	DNS Server	4+ octets	Adresse(s) des serveurs DNS
255	End	0 octet	Marque la fin des options

Format d'une option : `[code][length][data...]`

---

3. Machine à états minimale

```
INIT → SELECTING → REQUESTING → BOUND
  ↑                              ↓
  └──────────────── RENEWING ←──────────────┘
```

- **INIT** : Envoi DHCPDISCOVER en broadcast
- **SELECTING** : Attente d'un DHCPOFFER
- **REQUESTING** : Envoi DHCPREQUEST pour l'IP proposée
- **BOUND** : Configuration appliquée, attente du renouvellement

---

4. Étapes d'implémentation

Étape 1 : Construction du DHCPDISCOVER

```c
// Ports UDP
#define DHCP_CLIENT_PORT 68
#define DHCP_SERVER_PORT 67

// Initialiser le paquet
memset(&pkt, 0, sizeof(dhcp_packet_t));
pkt.op = 1;             // BOOTREQUEST
```

```
pkt.htype = 1;          // Ethernet
pkt.hlen = 6;           // Longueur MAC
pkt.xid = random_xid();  // Générer un XID aléatoire
pkt.flags = htons(0x8000); // Broadcast flag
memcpy(pkt.chaddr, mac_address, 6);
pkt.magic = htonl(0x63825363);

// Ajouter option 53 (Message Type = DISCOVER)
options[idx++] = 53;
options[idx++] = 1;
options[idx++] = 1;  // DHCPDISCOVER

// Ajouter option 255 (End)
options[idx++] = 255;
```

Envoyer en broadcast UDP vers `255.255.255.255:67` depuis `0.0.0.0:68`

Étape 2 : Réception du DHCPOFFER

- Vérifier que `op == 2` (BOOTREPLY)
- Vérifier que `xid` correspond
- Extraire `yiaddr` (votre future IP)
- Parser les options pour trouver :
  - Option 54 : Server Identifier (IP du serveur DHCP)
  - Option 51 : Lease time
  - Option 1 : Subnet mask
  - Option 3 : Gateway

Étape 3 : Envoi du DHCPREQUEST

Réutiliser la structure DHCPDISCOVER, mais :

```c
// Option 53 : Message Type = REQUEST
options[idx++] = 53;
options[idx++] = 1;
options[idx++] = 3;  // DHCPREQUEST

// Option 50 : Requested IP (yiaddr du OFFER)
options[idx++] = 50;
options[idx++] = 4;
memcpy(&options[idx], &offered_ip, 4);
```

idx += 4;

// Option 54 : Server Identifier (IP du serveur)
options[idx++] = 54;
options[idx++] = 4;
memcpy(&options[idx], &server_ip, 4);
idx += 4;

// End
options[idx++] = 255;
```

**Envoyer en broadcast vers `255.255.255.255:67`**

### Étape 4 : Réception du DHCPACK

- Vérifier `op == 2` et `xid`
- Vérifier option 53 == 5 (ACK)
- Configurer l'interface réseau avec :
  - IP : `yiaddr`
  - Masque : option 1
  - Gateway : option 3
  - DNS : option 6 (si besoin)

---

## 5. Gestion des timeouts

Implémentez des timeouts pour éviter les blocages :

- **DISCOVER** : Renvoyer après 4, 8, 16, 32 secondes (backoff exponentiel)
- **REQUEST** : Renvoyer après 4, 8, 16 secondes
- **Lease renewal** : Renouveler à T1 (typiquement 50% du lease time)

---

## 6. Fonctions utilitaires nécessaires

```c
// Parser une option DHCP spécifique
uint8_t* dhcp_find_option(uint8_t *options, int len, uint8_t code);

// Convertir endianness (si pas déjà fait)

```
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);

// Générer un XID aléatoire (utilisez un PRNG ou un timer)
uint32_t generate_xid(void);
```

---

7. Taille minimale des buffers

- **Buffer TX** : 300 octets suffisent (236 + ~60 options)
- **Buffer RX** : 576 octets minimum (MTU BOOTP/DHCP standard)

---

8. Considérations embarquées

- **RAM** : Structure statique, pas d'allocation dynamique
- **Stockage du lease** : Optionnel, mais permet de redemander la même IP au redémarrage (DHCPREQUEST avec option 50 dès INIT)
- **Renouvellement** : Implémenter un timer pour T1 (50% du lease) et T2 (87.5% du lease)
- **État minimal** : Sauvegarder uniquement IP, masque, gateway, serveur DHCP, XID, lease time

---

9. Séquence complète minimale

```
Client                          Serveur
  |                               |
  |-------- DHCPDISCOVER (broadcast) ------->|
  |                               |
  |<------- DHCPOFFER ----------------------|
  |                               |
  |-------- DHCPREQUEST (broadcast) -------->|
  |                               |
  |<------- DHCPACK ------------------------|
  |                               |
  [Configuration de l'interface réseau]
```

---

10. Code minimal de parsing d'options

```c
uint8_t* find_dhcp_option(uint8_t *opts, int len, uint8_t code) {
    int i = 0;
    while (i < len) {
        if (opts[i] == 255) break;  // End
        if (opts[i] == code) {
            return &opts[i];  // Retourne pointeur sur [code][len][data]
        }
        i += 2 + opts[i + 1];  // Skip code + len + data
    }
    return NULL;
}
```

Points de validation

Avant de considérer votre implémentation complète, vérifiez :

- ✓ Le XID est identique dans DISCOVER/REQUEST et vérifié dans OFFER/ACK
- ✓ Le magic cookie (0x63825363) est présent et correct
- ✓ Les adresses IP sont en network byte order (big-endian)
- ✓ Le broadcast flag (0x8000) est activé si vous ne pouvez pas recevoir d'unicast avant configuration
- ✓ L'adresse MAC est correctement copiée dans `chaddr`
- ✓ Les options incluent toujours le type de message (option 53)
- ✓ La fin des options est marquée par l'option 255

Note importante

Ce guide couvre l'essentiel pour un client DHCP fonctionnel. Pour un environnement de production, ajoutez la gestion des erreurs, le renouvellement automatique et éventuellement le support DHCPNAK.

Document créé pour l'implémentation d'un client DHCP en C embarqué
Format : Markdown - Facilement convertible en PDF, HTML ou autre format