République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Djillali Liabès - Sidi Bel Abbès

Cahier des Charges et Conception Technique pour mémoire de licence

pour l'obtention du diplôme de **Licence académique**

Domaine d'étude: **Informatique**

Spécialité: **Ingénierie des systèmes d'information et du logiciel**

Thème

Conception et Implémentation d'un Protocole d'Allocation Dynamique d'Adresses IP Inspiré de DHCP : Une Approche Innovante Utilisant Rust

---

Présenté par:
DIB Abdelkrim Yassine Taki-Eddine

Soutenu le: **Mai 2025**

Année universitaire: 2024/2025

Contents

0.1 Objectifs du projet

- Développer un protocole pour l'allocation dynamique d'adresses IP, inspiré de DHCP.
- Garantir la sécurité contre les attaques telles que les attaques de saturation DHCP.
- Implémenter le projet en utilisant le langage Rust pour sa sécurité et sa rapidité.

0.2 Fonctionnalités principales

0.2.1 Côté Client

- Envoyer une demande d'adresse IP au serveur.
- Recevoir et configurer l'adresse IP attribuée.
- Renouveler l'adresse IP avant expiration.

0.2.2 Côté Serveur

- Attribuer des adresses IP aux clients.
- Gérer une base de données des adresses utilisées.
- Implémenter des mécanismes de sécurité pour bloquer les comportements malveillants.

0.2.3 Communication Réseau

- Utilisation du protocole UDP pour la communication Client-Serveur.
- Définition de formats de messages clairs (requêtes, réponses, confirmations).

0.2.4 Sécurité

- Implémenter un système de limitation de débit (Rate Limiting).
- Ajouter une liste noire pour bloquer les clients malveillants.

0.3 Contraintes techniques

- **Langage de programmation :** Rust.
- **Protocole réseau :** UDP.
- **Base de données :** SQLite pour stocker les adresses et les journaux.
- **Bibliothèques utilisées :**
  - `tokio` : gestion asynchrone et sockets UDP.
  - `rusqlite` : gestion de SQLite.
  - `serde` : sérialisation/désérialisation des messages.

0.4 Organisation du projet

1. Apprendre Rust et configurer l'environnement.
2. Créer un prototype de client pour envoyer des messages au serveur.
3. Créer un prototype de serveur pour gérer les messages et répondre.
4. Ajouter les fonctionnalités de base (attribution IP, base de données).
5. Implémenter les mécanismes de sécurité.
6. Tester et optimiser le système.

0.5 Période nécessaire pour apprendre Rust et ressources

0.5.1 Durée estimée

Une heure par jour pendant **3 semaines** suffira pour maîtriser les bases nécessaires au projet.

0.5.2 Ressources recommandées

- **Livre officiel :** The Rust Programming Language (The Book).
- **Documentation officielle :** Rust Documentation.
- **Tutoriels vidéo :** Recherche "Rust Programming" sur YouTube (exemple : chaîne *"Code Academy"*).
- **Plateformes interactives :** Exercism (Rust Track).

0.6 Design technique

0.6.1 Architecture globale

Le projet est divisé en deux parties principales : **client** et **serveur**. Chaque partie est organisée en modules pour simplifier le code et faciliter les tests.

0.6.2 Structure du projet

```
project/
 src/
    main.rs          // Point d'entrée du projet
    client/          // Code client
       mod.rs       // Module principal du client
       discovery.rs // Découverte du serveur
       renewal.rs   // Renouvellement d'adresse IP
    server/          // Code serveur
       mod.rs       // Module principal du serveur
```

```
      allocation.rs // Allocation d'adresses IP
      database.rs // Gestion de la base de données
      security.rs // Sécurité (rate limiting, blacklist)
   message/        // Gestion des messages UDP
      mod.rs      // Module des messages
      serializer.rs // Sérialisation des messages
      deserializer.rs // Désérialisation des messages
   tests/          // Tests unitaires et fonctionnels
      client_tests.rs
      server_tests.rs
Cargo.toml          // Fichier de configuration Cargo
```

0.6.3 Modules et fonctionnalités

1. Client :

- Découvrir le serveur.
- Demander une adresse IP.
- Renouveler l'adresse IP avant expiration.

2. Serveur :

- Répondre aux demandes de découverte.
- Attribuer des adresses IP disponibles.
- Enregistrer les adresses attribuées dans une base de données.

3. Sécurité :

- Limiter le débit de requêtes par client.
- Bloquer les clients malveillants via une liste noire.

4. Messages UDP :

- Sérialiser et désérialiser les messages pour les échanges réseau.
- Définir les formats pour les messages (découverte, offre, demande, confirmation).

0.7 Estimation des délais

Étape	**Description**	**Durée estimée**
Apprentissage Rust	Comprendre les bases du langage et configurer l'environnement.	3 semaines
Prototype client	Envoyer des messages UDP au serveur.	1 semaine
Prototype serveur	Recevoir et répondre aux messages UDP.	1 semaine
Gestion des adresses	Ajouter la base de données et gérer les allocations IP.	1 semaine
Ajout de sécurité	Implémenter les mécanismes de sécurité.	1 semaine
Tests complets	Tests unitaires et fonctionnels pour assurer la fiabilité.	1 semaine
Documentation	Rédaction de la documentation technique.	1 semaine