Système de Connexion Pair-à-Pair en Half Duplex

1 Introduction

Ce rapport présente le sous-système de connexion entre deux appareils développé en Python. Le système utilise une architecture client-serveur avec un mécanisme de codage personnalisé pour faciliter l'établissement de la connexion.

2 Architecture Générale

Le système se compose de deux composants principaux :
— **server.py** : Application serveur qui écoute les connexions entrantes
— **client.py** : Application cliente qui initie la connexion au serveur

3 Mécanisme de Connexion

3.1 Détection d'Adresse IP

Le serveur utilise la fonction `get_local_ip()` pour déterminer automatiquement son adresse IP locale :
— Priorité aux interfaces réseau (via `netifaces`)
— Fallback sur l'adresse de l'hôte local
— Filtrage des adresses privées (192.168.*, 10.*)

3.2 Système de Codage de Connexion

3.2.1 Base 64 Personnalisée

Une base 64 personnalisée est définie :
`"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz!?"`

3.2.2 Encodage (Côté Serveur)

La fonction `encode_connexion_code()` transforme l'IP et le port en un code de 8 caractères :

1. Conversion de l'IP en quatre segments binaires de 8 bits
2. Conversion du port en binaire sur 16 bits
3. Concatenation en une chaîne binaire de 48 bits
4. Découpage en 8 segments de 6 bits
5. Conversion de chaque segment en caractère base64

3.2.3 Décodage (Côté Client)

La fonction `decode_connexion_code()` effectue l'opération inverse :

1. Conversion de chaque caractère en valeur binaire 6 bits
2. Reconstruction de la chaîne binaire complète (48 bits)
3. Extraction des composants IP (32 bits) et port (16 bits)
4. Conversion en format IP décimal et port entier

4 Protocole de Communication

4.1 Établissement de Connexion

1. Le serveur démarre et affiche le code de connexion
2. Le client décode le code pour obtenir IP et port
3. Connexion TCP standard via `socket.connect()`
4. Message de bienvenue envoyé par le serveur

4.2 Système de Chat

Le protocole de communication suit un modèle half-duplex :
— Alternance stricte des messages client-serveur
— Taille de buffer fixe à 1024 octets
— Gestion basique de fin de session via intérruption clavier

5 Avantages du Système

5.1 Simplicité d'Utilisation

— Code de connexion facile à communiquer (8 caractères)
— Détection automatique de l'adresse réseau texte simple et intuitive

5.2 Robustesse

— Gestion des erreurs de connexion
— Fallback en cas d'échec de détection d'IP
— Fermeture propre des sockets

5.3 Portabilité

— Utilisation de bibliothèques Python standard
— Compatible avec différentes plateformes
— Indépendant de la configuration réseau spécifique

6 Limitations Identifiées

— Dépendance optionnelle à `netifaces` (installation requise)
— Gestion basique des erreurs réseau
— Code de connexion limité aux adresses IPv4