DAN MBWEY BIZWA
M1 DFS

Rapport projet TCP-IP

Le projet consiste à développer une application Clients/Serveur qui permet à des clients de récupérer des fichiers enregistrés sur le serveur en utilisant le protocole de communication sous TCP-IP.

Pour ce projet nous avons créé 2 fichiers python au nom de ftpClient.py pour gérer la connexion du client vers le server afin de récupérer des fichiers et ftpServer.py qui affiche la liste des fichiers disponible dans le server que l'utilisateur peut récupérer.

A. Partie Server

Ci-dessous sont les différentes étapes essentielles pour la réalisation du server :

1. Nous avons commencé par l'importation de librairies telles que **OS** pour pouvoir lister les noms de différents fichiers se trouvant dans le dossier et aussi pour créer des dossiers s'ils n'existent pas, **SOCKET** pour pouvoir lancer le server, et **JSON** afin de pouvoir convertir le dictionnaire qui devra être envoyer au client en cas de téléchargement d'un fichier. Ensuite, la déclaration et définition des variables HOST, PORT et BUFSIZ (buffer size)
2. La création d'une classe **node** pour nous aider dans la construction du Huffman Tree.
3. La création de différentes fonctions pour la lecture du fichier à envoyer et la construction du Huffman Tree telles que :
   - readContent() : pour la lecture du contenu d'un fichier
   - frequenceCalc() : Calcule la fréquence des lettres dans le contenu venant d'un fichier
   - printNodes() : Pour le remplissage du dictionnaire ayant comme **key** les lettres se trouvant dans le contenu du fichier et comme **value** les codes en binaire produit par l'algorithme de Huffman.
   - encodage() : Remplace toutes les lettres du contenu d'un fichier par leurs correspondances en code binaire produit par l'algorithme de Huffman.(Compression)
   - huffman() : construit le Huffman Tree. En utilisant les différentes lettres se trouvant dans le fichier.
   - runServer() : lance le server.

B. Partie Client

Ci-dessous sont les différentes étapes essentielles pour la réalisation du client :

1. Nous avons commencé par l'importation de librairies telles que **OS** pour pouvoir lister les noms de différents fichiers se trouvant dans le dossier et aussi pour créer des dossiers s'ils n'existent pas, **SOCKET** pour pouvoir lancer le server, et **JSON** afin de pouvoir convertir le dictionnaire qui devra être envoyer au client en cas de téléchargement d'un fichier. Ensuite, la déclaration et définition des variables HOST, PORT et BUFSIZ (buffer size)
2. La création de différentes fonctions pour le décodage du message envoyer en forme binaire et le lancement du client :
   - decodage() : Décode le message reçu du server. (Décompression)
   - runClient() : lance le client.

C. Fonctionnement du programme

DAN MBWEY BIZWA
M1 DFS

1. Placer les 2 fichiers python dans un même dossier. (Pour un accès plus rapide au terminal)
2. **Créer un dossier au nom de myfiles dans le même dossier que 2 fichiers python. Ensuite placer les fichiers en format .txt que le client verra pour télécharger. Le programme ne fonctionne qu'avec les fichiers .txt pour l'instant mais il peut être adapter pour fonctionner avec d'autres types de fichiers.**
3. Lancer le server à partir du terminal

```
PS C:\Users\DMB\Documents\ftp> python ftpserver.py
Server waiting for connection...
```

4. Lancer la partie client à partir d'un autre terminal : après sa connexion sur le server, il lui présente la liste des fichiers pouvant être télécharger et demande au client saisir le numéro du fichier qu'il souhaite télécharger. **Dans la partie client, le numéro du PORT et HOST du server sont définis manuellement.**

```
PS C:\Users\DMB\Documents\ftp> python ftpclient.py
Files to be downloaded

1. Dan_cover_letter.pdf
2. data.txt
3. hesap.txt
4. Projet-TCP-IP.doc
5. TP NO SQL.docx

Enter document number to download:
```

5. Après la connexion du client, dans le terminal du server nous avons la notification du client qui vient de se connecter.

```
PS C:\Users\DMB\Documents\ftp> python ftpserver.py
Server waiting for connection...
Client connected from:  ('127.0.0.1', 52688)
```

6. Si le client saisi un numéro qui n'existe pas dans la liste, le server lui renvoi un message d'erreur et lui demande s'il veut continuer ou pas.

```
PS C:\Users\DMB\Documents\ftp> python ftpclient.py
Files to be downloaded

1. Dan_cover_letter.pdf
2. data.txt
3. hesap.txt
4. Projet-TCP-IP.doc
5. TP NO SQL.docx

Enter document number to download: 0
Incorrect value
Do you Want to download more files from the server[y/n]:y
Enter document number to download: 9
Incorrect value
Do you Want to download more files from the server[y/n]:y
Enter document number to download:
```

7. Après la saisie du correcte numéro :

- Le server cherche le fichier dans son répertoire
- Fait la lecture de son contenu
- Compresse à l'aide de l'algorithme de Huffman
- Envoi le contenu compressé et le dictionnaire
- Du côté client on reçoit ce petit résumé de ce qui vient de se passer et un dossier au nom de **Received** sera automatiquement créé contenant le fichier décompressé reçu.

```
Do you Want to download more files from the server[y/n]:y
Enter document number to download: 2
The selected file is data.txt
The encoded message is 11000011100010000011111100110010011101011110101011011111001101010110
11100111111011000000010111110100 10010
The dictionary is {"i": "0000", "d": "0001", "o": "001", ".": "010", "e": "011", "l": "1000
", "r": "1001", "t": "1010", "h": "1011", "H": "11000", "f": "11001", "m": "11010", "s": "1
1011", " ": "111"}
The decoded message is :  Hello from the other side ...
Do you Want to download more files from the server[y/n]:
```

- En saisissant **n** l'exécution du client s'arrête.

D. Conclusion

Le programme fonctionne correctement sans problème en produisant le résultat attendu. Cependant il peut être amélioré en incluant le soutien de plusieurs types de fichiers et une interface utilisateur.