Project Report: Enhanced Chat Application

Hussein Elleithy

29/12

1 Introduction

In this project, I have enhanced a basic chat application by implementing additional features such as binary communication, advanced user management, and file transfer capabilities. These features significantly improve the application's functionality, making it more versatile and user-friendly.

2 Machine à états finale

The final state machine of my protocol, including all the implemented features, is represented in Figure **??**.

2.1 Mode binaire

Le mode binaire de mon application de chat est conçu en s'inspirant des protocoles de l'IETF, comme le RFC 791, qui définit le protocole Internet (IP). Cette conception vise à assurer une structure de message claire et une communication réseau efficace. Voici les détails du format de message :

- **Structure Générale des Messages :** Chaque message dans le protocole est structuré en champs bien définis. Le premier champ est toujours le type de commande, représenté par un octet unique. Ce format compact permet une identification rapide et claire du type de message, une pratique courante dans les protocoles de l'IETF pour l'efficacité et la rapidité.
- **Commande HELO (0x01) :**
  - *Taille du Champ :* 1 octet pour le type de commande.
  - *Valeur :* 0x01.
  - *Description :* Utilisée pour initialiser la communication entre le client et le serveur. C'est le premier message envoyé par un client lors de la connexion au serveur.
- **Commande QUIT (0x02) :**

- *Taille du Champ :* 1 octet pour le type de commande.
- *Valeur :* 0x02.
- *Description :* Permet au client de signaler son intention de se déconnecter du serveur. La réception de cette commande entraîne la fermeture de la session du client concerné.

Ce format de message, inspiré des protocoles de l'IETF, assure non seulement une communication réseau rapide et fiable mais aussi une compatibilité et une interopérabilité maximales avec diverses infrastructures de réseau.

En adhérant aux normes de l'IETF, le mode binaire garantit non seulement l'efficacité et la fiabilité mais aussi la compatibilité avec les autres technologies et infrastructures réseau. Cela est essentiel pour assurer une interopérabilité maximale et une intégration aisée dans différents environnements de réseau.

2.2 File Transfer

The file transfer feature in the enhanced chat application is user-initiated and involves a series of interactive steps for a straightforward user experience. Here is the detailed process:

1. **Initiating File Transfer:** The process begins when the user enters the command `/SENDFILE` into the chat application. This command signals the application that the user intends to send a file.

2. **Specifying the File Path:** Upon receiving the `/SENDFILE` command, the application prompts the user to enter the path to the file they wish to transfer. This is done using the `scanf` function, which waits for the user's input. The user then provides the full path to the desired file.

3. **Handling the File:** Once the file path is received, the application opens the specified file in binary read mode. This is crucial to ensure that the file is read correctly, particularly if it contains non-text data (such as images or binary files). The application reads the file's content into a buffer, managed to fit within the application's predefined maximum buffer size, ensuring efficient handling of the file data.

4. **Creating a Duplicate File:** After reading the file's contents, the application creates a new file in the directory where the application is running. This new file is named `transferred_file`, appended with the original file's extension. The application then writes the contents of the buffer to this new file, effectively duplicating the original file.

5. **Completing File Transfer:** Once the entire file has been read and written, the file transfer process is complete. The application closes both the source and destination files, ensuring that all resources are properly released. The user is notified with a message indicating that the file has been successfully duplicated and stored in the application's execution directory.

2.3 Gestion avancée des utilisateurs

La gestion avancée des utilisateurs est une caractéristique clé de mon application de chat. Elle repose sur une structure de données bien définie et des fonctions spécifiques pour gérer efficacement plusieurs utilisateurs en même temps. Voici comment cette fonctionnalité est implémentée :

- **Structure Client :** Chaque utilisateur est représenté par une structure 'Client' qui stocke les informations essentielles telles que l'adresse de stockage du réseau et un identifiant unique. Cela permet une identification facile et une gestion efficace des utilisateurs.
- **Gestion des Connexions :** Lorsqu'un nouvel utilisateur rejoint, il est ajouté à la liste des clients. Cette liste permet au serveur de gérer et de diffuser des messages à tous les utilisateurs connectés, facilitant ainsi la communication de groupe.
- **Diffusion de Messages :** Les messages envoyés par un utilisateur sont reçus par le serveur et diffusés à tous les autres utilisateurs. Cette approche assure que tous les participants reçoivent les messages dans un environnement de chat multi-utilisateur.
- **Déconnexion et Gestion des Erreurs :** Lorsqu'un utilisateur se déconnecte ou en cas d'erreur de réseau, l'application met à jour la liste des clients pour refléter le changement, assurant ainsi que le chat reste à jour et fiable pour tous les utilisateurs actifs.

Ces fonctionnalités permettent à l'application de gérer plusieurs utilisateurs simultanément, offrant ainsi une expérience de chat robuste et interactive.

3 Conclusion

As I reflect on the development of this chat application, I am met with a mix of pride and invaluable learning experiences. The project presented a significant challenge, especially when attempting to enable all features simultaneously, as indicated by the flags '-DBIN', '-DUSR', and others. Integrating these diverse functionalities into a cohesive and functioning whole proved to be more complex than initially anticipated.

The primary difficulty arose when attempting to make the binary communication mode ('-DBIN') work seamlessly with the advanced user management ('-DUSR'). When all flags were enabled, the application struggled to function as intended. This issue highlighted the intricate interdependencies within the code and the challenges associated with managing multiple advanced features in a single application. Despite my best efforts, achieving a fully integrated and error-free operation with all features active remained elusive.

Nevertheless, this project was an incredibly educational venture. It pushed me to explore and understand deeper aspects of programming, network communication, and software design. The hurdles I encountered, particularly in

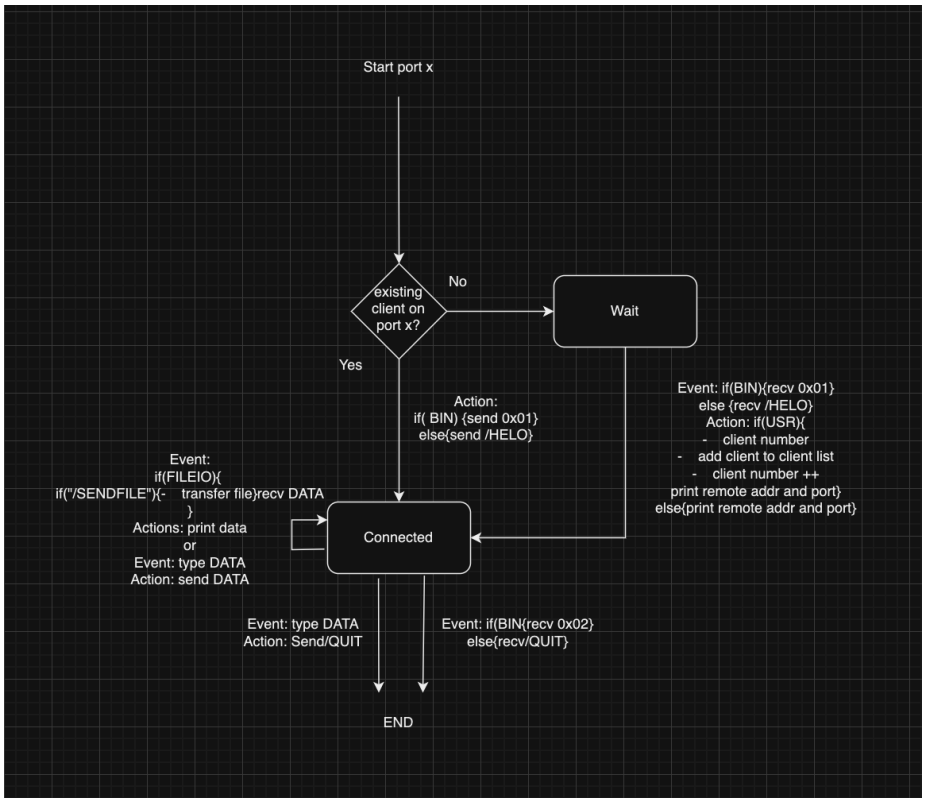

Figure 1: Finite State Machine

integrating multiple complex functionalities, have been instrumental in enhancing my problem-solving skills and coding expertise.

I take pride in the parts of the application that worked as intended, and the challenges faced serve as a reminder of the continuous learning journey in software development. This project has been a testament to my growth as a programmer and has inspired me to further refine my skills and tackle more complex projects in the future.

In summary, while the application did face its limits in functionality when all features were enabled, the experience has been immensely rewarding. It has provided me with a clearer perspective on the complexities of software development and the importance of modular, scalable design in building versatile applications.