Universidade Federal Fluminense

Instituto de Computação

Graduação em Ciência da Computação

Discentes

Valesca Moura de Sousa

Victor Faria Fernandes

Victor Miranda Flórido

Trabalho Prático de Redes de Computadores II

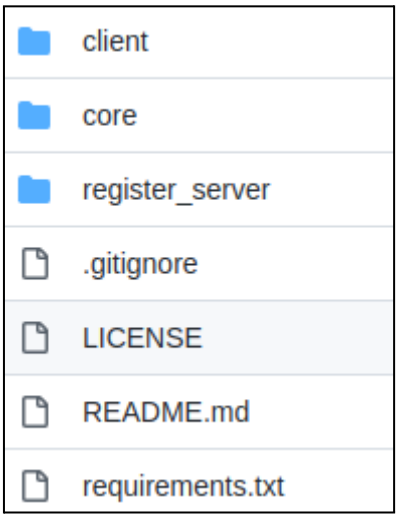
2022-2

Desenvolvimento de uma aplicação baseada no paradigma cliente-servidor utilizando socket de rede.

Novembro/2022

IMPLEMENTAÇÃO

Link do GitHub: https://github.com/valescamoura/cliente-servidor-redesII-20222.



Estrutura do código-fonte

A aplicação, desenvolvida na linguagem Python, se trata de uma espécie de chat de voz com uso de servidores de voz e de registro de usuários onde a comunicação entre cliente e servidor é realizada por meio de sockets. No caso do servidor de registro, o cliente inicia uma conexão TCP que permanece até que o usuário saia da “sala de voz” e remova seu registro. No caso do servidor de voz, o socket utilizado para enviar as mensagens de voz é o UDP. A aplicação utiliza a biblioteca PyAudio e as bibliotecas socket e _thread, nativas do Python. Ela está dividida em três partes: core, register_server e client. Os tópicos a seguir explicam cada uma das partes em mais detalhes. A fim de facilitar o entendimento do código-fonte foram adotadas as melhores práticas de programação padronizadas para o Python. **Ademais, detalhes mais específicos em relação a parâmetros de funções podem ser vistos em comentários no código.**

Core

Definição de classes e funções genéricas a serem utilizadas tanto pelo cliente quanto pelos servidores de voz e de registro. Nesta parte são definidas as classes de manipulação e estruturação para tratamento de áudio, requisições e formato do usuário. Além disso, definimos uma classe genérica para o servidor que pode ser utilizada para ambos os servidores: de voz e de registro. Para tal, a classe permite trabalhar tanto com socket TCP, para o servidor de registro quanto uso de sockets UDP para os servidores de voz, isso é determinado através do parâmetro *protocol* do construtor da classe *Server* ao instanciar o servidor. Ademais, a classe *Server* permite instanciar servidores multithreading ou com uma única thread. A opção multithreading foi implementada devido a necessidade do servidor de registro precisar manter conexão TCP com mais de um cliente simultaneamente, visto que após o cliente se conectar ao servidor de registro, ele só encerra sua conexão ao apagar seu registro do servidor. Para isso, o servidor TCP espera que o cliente inicie uma conexão e assim que a conexão é estabelecida ele a passa para outra thread e continua esperando por novas conexões. Devido à classe *Server* ser genérica, ao instanciar o servidor é passada a função que o servidor executará ao receber

uma mensagem. No caso do servidor de registros, é uma função de manipulação dos registros de usuário, enquanto no servidor de voz se trata de uma função da manipulação de áudio.

Register Server

Nesta parte é feita a definição da classe *Register* que guarda e manipula a tabela de usuários registrados no serviço de voz. Além disso, existe o código principal do servidor de registro, que é uma instância da classe genérica Server utilizando sockets TCP e multithreading, além da definição da função da *client_usecase* executada pelo servidor. Nesta função são implementadas as regras de negócio do servidor de registro com a definição das operações suportadas e seus respectivos métodos e respostas.

Client

Nesta seção foi realizada a implementação do cliente do servidor de registro e do servidor de voz, além do servidor de voz propriamente dito, que se trata de uma instância da classe *Server* utilizando sockets UDP e definindo a regra de negócio a ser executada pelo servidor de voz.

USO DA APLICAÇÃO

Para a utilização da aplicação é necessário de início instalar os pré-requisitos do sistema, sendo esses o Python 3 e o PyAudio. Em seguida será essencial configurar o PYTHONPATH apontando para a raiz do projeto e executar o script *run.sh* utilizando o terminal, configurar a permissão para rodar o mesmo e depois executar o ***./run.sh register_server***, para rodar o servidor de registros, ou o ***./run.sh client*** para rodar o cliente/servidor de voz.

Depois disso, basta ler o output e seguir as instruções via terminal.