Relatório sobre Protocolos de Comunicação

Nycolas Felix, Rebecca Simão, Renan Gomes, Afonso Mateus, Filipe Viana
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil
nycsfelix@poli.ufrj.br, rebecca.gomsim@poli.ufrj.br, rencvlg@poli.ufrj.br, afonsomateus@poli.ufrj.br, filipe.vianasilva@poli.ufrj.br

Resumo. *Este relatório documenta o projeto colaborativo referente ao primeiro trabalho parcial para a disciplina de Redes de Computadores I. Em turma, os alunos foram requisitados a criação de um protocolo comum a todos da turma. E, em grupo, deveriam criar um cliente e servidor deste protocolo, mais um cliente/servidor P2P. Os documentos adicionais tais quais o documento para definição do protocolo, bem como o código fonte foram anexados juntamente ao envio deste relatório.*

Palavras-chave: *Redes de computadores, Protocolos, P2P, sockets*

I. INTRODUÇÃO

Este documento apresenta instruções detalhadas para a criação de um cliente, onde teremos um espaço para bate-papo entre clientes diferentes dentro de um servidor. Ambos, o servidor e os clientes, devem ser capazes de se conectarem e desconectarem mutuamente, e do mesmo modo receber e enviar mensagens.

O objetivo desse estudo é aplicar os conhecimentos relativos ao desenvolvimento de protocolos - orientado ao tipo de conexão TCP, para que seja promovido um melhor entendimento dos estudantes acerca desse tema na prática.

II. METODOLOGIA

A ideia geral para o desenvolvimento das linhas de código foi a criação de um chat de comunicação P2P mantido por um servidor para a comunicação entre clientes que se conectem ao chat. Para iniciar essa conexão com o servidor recorremos ao endereço IP, mais a adição da porta específica escolhida.

O chat é utilizado para a troca de mensagens de texto (padrão utf-8) e utiliza um protocolo TCP para a comunicação entre os sockets dos clientes com o socket do servidor. Ele funciona em base de um sistema de espelhamento de mensagens,

As ferramentas principais utilizadas neste projeto são constituídas principalmente de dois arquivos escritos em Python, ClienteSocket.py e ServidorSocket.py. Para a execução do código, foi utilizado o software de programação Visual Studio Code pelos membros do grupo, a fim de ser compilado e cumprir o objetivo do trabalho mostrado.

Foi utilizado o protocolo TCP ao invés de UDP, por suas vantagens ao atenuar problemas recorrentes como o congestionamento de dados e o descontrole do fluxo dessas mensagens. Vantagens essas que auxiliam no desempenho do chat.

Ademais, foi escolhida a mensagem ":D" para afirmar desconexão para o usuário. Este acréscimo possibilita maior entendimento ao se utilizar do chat. Contudo, embora também exista uma mensagem afirmando conexão ao servidor, esta não é visível ao cliente. Prejudicando possivelmente o entendimento deste para com o projeto - o que poderia ser uma melhoria futura para o protocolo criado.

O projeto é tolerante ao atraso, assim possibilitando que mensagens enviadas possam não chegar de forma instantânea, sem serem perdidas no caminho. Dessa forma, não possui funcionamento em tempo real. Para mais, ele é também tolerante a perda dado que o protocolo TCP é o responsável por lidar com essa questão.

III. DESENVOLVIMENTO

Inicialmente foi verificado o conhecimento adquirido em sala a respeito do funcionamento da comunicação TCP/IP e sockets, sendo implementado um chat público específico da turma que troca mensagens de texto no formato 'utf-8'. Para tal, utiliza-se um protocolo feito de autoria da própria turma. O cliente, representado por um computador executando o programa ClienteSocket.py, consegue enviar mensagens para todos os outros Clientes conectados no Servidor de forma assíncrona, contanto que a mensagem não exceda 64 caracteres (bytes) de tamanho.

O desenvolvimento do Cliente foi feito seguindo as características do protocolo definidas pelo servidor (utilização do TCP, encodificação das mensagens em UTF-8, tamanho em quantidade de bytes para as mensagens trocadas, porta utilizada etc).

A partir dessas características obtidas do servidor, a implementação foi feita utilizando as bibliotecas nativas do python, socket e threading.

- **Socket**:
  A biblioteca socket é uma biblioteca do python que permite que nós criemos um socket para que um cliente ou servidor possam se comunicar um com o outro. No socket criado, podemos especificar a versão do IP como IPv4 ou IPv6 e podemos definir também o protocolo de transporte utilizado, TCP ou UDP. Os escolhidos para o caso do nosso cliente seguem o acordo com o criador do servidor em que foi usado o IPv4 e o TCP que a turma, em conjunto, julgou ser o ideal. A partir disso, o servidor

e os clientes são criados com e ssas configurações, o servidor é vinculado a um endereço e uma porta e o cliente se conecta a ele por meio desse mesmo endereço e porta.

- **Threading**:
  A biblioteca threading é utilizada pra realizar multi-threading no código, isto é, para podermos no mesmo código executar diferentes operações em paralelo. No caso do servidor, este roda uma função (start()) que coloca o servidor em estado de leitura (aberto a conexão de clientes) e inicia um loop em que verifica e aceita a conexão dos clientes. Para cada um dos clientes conectados, é criada uma thread própria que é uma função que irá tratar do recebimento de mensagens desse cliente e irá replicar cada mensagem que ele enviar para todos os outros clientes conectados, permitindo o funcionamento da aplicação chat.
  Para o cliente, após se conectar ao servidor/chat, possui duas funções rodando em paralelo, a função enviarMensagem(), que recebe mensagens escritas no terminal pelo cliente e envia ao servidor e a função receberMensagem() que serve para receber as mensagens enviadas pelos outros clientes e transmitidas pelo servidor paara todos os demais clientes.

IV. RESULTADOS E DISCUSSÕES

A comunicação ocorreu conforme o objetivo: o cliente envia a mensagem para o servidor, onde todos os outros clientes conectados conseguem vê-la e enviar suas mensagens próprias, com certas restrições: as mensagens não podem exceder 64 caracteres, conter acentos, crases, cedilhas, e outros sinais que não podem ser codificados pelo sistema 'utf-8'.

Por mais que a comunicação exceda no que se comprometeu, foi verificado que o sistema não dispensa melhorias. Primeiramente, do jeito que está escrito atualmente, o sistema diferencia diferentes usuários apenas pela porta sendo usada na comunicação, em vez de um usuário único. Assim, implementar uma devida interface no arquivo de cliente ajudaria a manter a comunicação mais clara para outros usuários. Outra mudança poderia ser o uso de um outro protocolo de codificação de texto que não o "utf-8", já que este não suporta acentos, crases, e outros sinais de escrita presentes na língua portuguesa. Permitindo uma melhor comunicação entre as partes.

V. CONCLUSÕES

Com a chegada das mensagens no chat, foi demonstrado um exemplo de comunicação P2P entre os clientes conectados no servidor. Nesse tipo de comunicação, um sistema final qualquer - como um computador, é configurado como cliente e servidor, e pode trocar informações diretamente com outros sistemas que o endereçarem a partir do endereço IP e da porta usada. Esta implementação do chat é apenas uma das muitas possibilidades de sistemas onde esse tipo de comunicação pode ser implementada, e essa foi escolhida especialmente pela turma por ser relativamente fácil de implementar, testar e corrigir em caso de falhas.

REFERENCES

[1] G. McMillan,"HOWTO sobre a Programação de Soquetes ",https://docs.python.org/pt-br/3/howto/sockets.html
[2] V. Kishlay, "Checkbox HTML: Socket Programming in Python",https://www.geeksforgeeks.org/socket-programming-python/?ref=gcse/