

Protocolo em Camada de Transporte da Rede

Felipe Costa¹, Gabriel Viana Thomaz¹, Mateus Nascimento Barbosa¹, Pedro Dionísio¹, Rodrigo Ferreira Bostrom¹, Victor De Luca S. N. Silva¹

¹Escola Politécnica – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{bc.felipe05,gv.thomaz,mateusx,pedrodionisio30,rodrigo_bst,vicdlsns}@poli.ufrj.br

Abstract: *This project aims to create two communication protocols, one being client-server and the other P2P, in which the exchanged messages in a session are saved in a text file.*

Resumo: *Este projeto tem como objetivo criar dois protocolos de comunicação, um cliente-servidor e outro P2P, nos quais as mensagens trocadas em uma sessão são salvas em um arquivo de texto.*

1. Introdução

O projeto descrito neste documento visa o desenvolvimento de um protocolo de comunicação baseado no modelo cliente-servidor, no qual uma mensagem enviada pelo cliente é recebida e interpretada por um servidor central ativo, o qual emite uma resposta. Tendo em vista a conexão estabelecida, tem-se como objeto final as mensagens destinadas a este cliente.

Além deste, apresentamos também neste projeto o desenvolvimento de um protocolo P2P, com o objetivo de implementar uma comunicação entre diferentes dispositivos através da formação de uma arquitetura de conexão descentralizada.

2. Descrição

O protocolo cliente-servidor construído apresenta características majoritárias que se assemelham a transmissão UDP. Isto se dá pelo fato de que não foram utilizados alguns mecanismos que auxiliam na confirmação do recebimento da mensagem. Além disto, podemos citar outras características relacionadas ao UDP como a ausência de controle de conexão e a não garantia de entrega de dados na ordem em que foi enviada.

Já o protocolo P2P é caracterizado pela descentralização das funções da rede, onde cada nó realiza tanto funções de servidor como de cliente, permitindo assim o compartilhamento de serviços e dados sem a necessidade de um servidor central. Apresenta uma maior disponibilidade dos dados pois os objetos podem ser disponibilizados em inúmeros nós porém a garantia de segurança é inferior aos outros tipos de projeto de compartilhamento de dados.

2.1. Sintaxe

Para o projeto foi utilizado a linguagem Python versão 3.7.4 com uma implementação orientada a objetos. Basicamente, devem ser instanciados objetos das classes correspondentes, que podem ser Server, Client ou P2P, e chamada a função *run()* para iniciá-los. Nesta implementação, quando o programa for executado o objeto correspondente já será instanciado e inicializado por padrão.

Para abrir um Server, é necessário apenas instanciar a classe e chamar a função *run()* e então este começará a receber conexões, enquanto que para abrir um Client deve-se instanciar a classe, chamar a função *run()* e após isto, o programa recebe do usuário um *username* e o endereço IP do Server, e, então, se conecta ao mesmo; após se conectar ao Server, o Client tem a possibilidade de digitar a mensagem a ser enviada, que será encaminhada aos demais Client's. O conteúdo da mensagem é guardado numa variável do tipo string e, além dela, são mostradas aos destinatários o *username* do remetente. Simultaneamente, as mensagens enviadas por outros Client's serão exibidas.

No caso P2P não há classes, mas apenas funções. São criadas threads em todas as máquinas envolvidas para que elas estejam abertas para a comunicação. Através do recurso Broadcast importado de Socket é possível realizar a troca de mensagens de forma P2P.

2.2. Escolhas de design

Após a definição do protocolo a ser utilizado pela turma, surgiu o desafio de como implementá-lo. O grupo escolheu por realizar a tarefa utilizando dos benefícios da programação orientada a objeto.

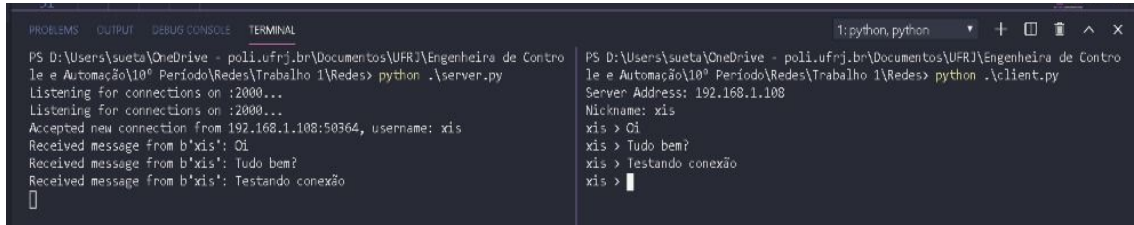
Para o modelo Cliente-Servidor foram criadas duas classes separadas, uma Client e outra Server, e dentro dessas classes definimos funções que determinam desde a porta a ser utilizada até a decodificação das mensagens enviadas, garantindo assim o funcionamento integral do protocolo.

Para o modelo P2P, não há uma classe característica mas apenas funções para executar os processos correspondentes; além disso, foi necessário o uso de threads para possibilitar a comunicação do modo correto.

3. Teste

Os testes de conexão executados no nosso projeto podem ser verificados através da análise das imagens da próxima seção. Na imagem 1, é apresentado o teste do modelo cliente-servidor, na imagem 3 do modelo P2P, e na imagem 2 é apresentada a interface do software Wireshark, que é um programa utilizado para a verificação do tráfego de rede. Durante o teste de comunicação utilizando os protocolos, o Wireshark capturou e analisou o tráfego originado pelos mesmos, comprovando, portanto, a sua funcionalidade.

4. Imagens



```
PS D:\Users\sueti\OneDrive - poli.ufrj.br\Documentos\UFRJ\Engenheira de Controle e Automação\10º Período\Redes\Trabalho 1\Redes> python .\server.py
Listening for connections on :2000...
Listening for connections on :2000...
Accepted new connection from 192.168.1.108:50364, username: xis
Received message from b'xis': Oi
Received message from b'xis': Tudo bem?
Received message from b'xis': Testando conexão
[]

PS D:\Users\sueti\OneDrive - poli.ufrj.br\Documentos\UFRJ\Engenheira de Controle e Automação\10º Período\Redes\Trabalho 1\Redes> python .\client.py
Server Address: 192.168.1.108
Nickname: xis
xis > Oi
xis > Tudo bem?
xis > Testando conexão
xis >
```

Imagem 1: Server e Client abertos

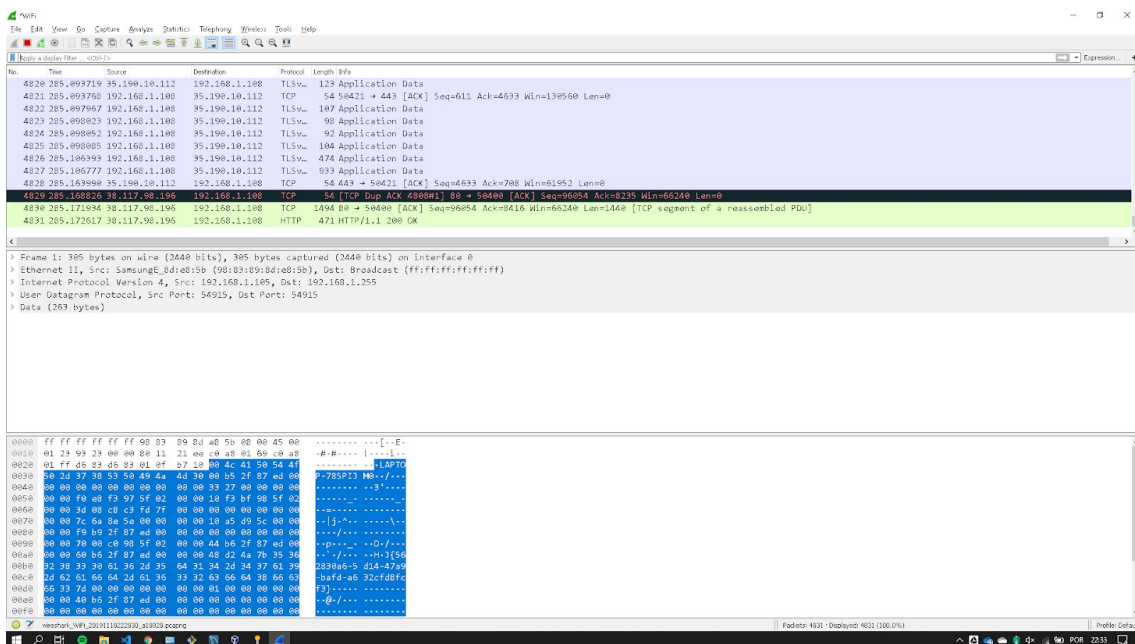
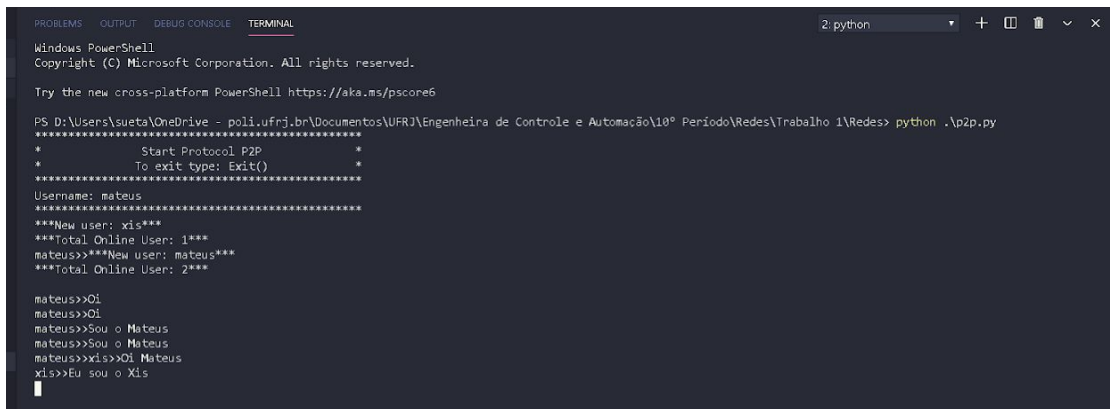


Imagem 2: Wireshark rastreando a rede



```
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Users\sueti\OneDrive - poli.ufrj.br\Documentos\UFRJ\Engenheira de Controle e Automação\10º Período\Redes\Trabalho 1\Redes> python .\p2p.py
*****
*          Start Protocol P2P          *
*          To exit type: Exit()        *
*****
Username: mateus
*****
***New user: xis***
***Total Online User: 1***
mateus>***New user: mateus***
***Total Online User: 2***

mateus>>Oi
mateus>>Oi
mateus>>Sou o Mateus
mateus>>Sou o Mateus
mateus>>xis>>Oi Mateus
xis>>Eu sou o Xis
```

Imagem 3: P2P aberto

5. Github

Os códigos e instruções se encontram no repositório <https://github.com/vicdlsns/Redes>.

6.Referencias

Python Software Foundation (2019) “socket — Low-level networking interface”,
<https://docs.python.org/3/library/socket.html>, Novembro.

Socket.IO “Overview - Broadcasting messages”,
<https://socket.io/docs/>, Novembro.

Real Python (2012-2019) “An Intro to Threading in Python”,
<https://realpython.com/intro-to-python-threading/>, Novembro.