ChatWF desenvolvimento do Cliente-IRC com interface – Projeto 1

Akin Sangiàcomo Bazila - 221002002, Guilherme Fornari Leonel - 221017023, Pedro Avila Beneveli - 221001972

***Resumo/abstract*—Foi realizada a implementação de um cliente IRC como parte de um projeto acadêmico de Redes de Computadores. O cliente foi desenvolvido em Python, empregando a biblioteca de sockets padrão e a biblioteca PyQt5 para a interface gráfica. O trabalho tem como objetivo promover a compreensão da camada de aplicação, de forma a explorar o desenvolvimento de aplicações que utilizam a internet e trocam mensagens, neste caso, o IRC.**

**Palavras-chave—aplicação, cliente, IRC**

1. Introdução

N

este artigo iremos documentar a implementação e o funcionamento do primeiro projeto da disciplina Redes de computadores, onde foi desenvolvido um Cliente IRC utilizando a linguagem de programação Python em sua versão atual, 3.11.6. Esse trabalho visa melhorar o aprendizado da camada de aplicação ao promover o entendimento de como funciona o desenvolvimento de aplicações que utilizam a Internet e de como essas aplicações enviam e recebem mensagens na rede a partir do socket.

Esse relatório está dividido em 5 seções. Temos a própria introdução, que contextualiza esse trabalho, a fundamentação teórica, que descreve conceitos e técnicas utilizadas para a compreensão e realização do trabalho, a análise experimental e análise de resultados, que irão descrever os resultados obtidos nesse trabalho, junto com alguns exemplos do funcionamento do cliente IRC desenvolvido, a conclusão, que apresenta as conclusões sobre os resultados e a aplicação desenvolvida, e a Bibliografia, que apresenta referências utilizadas durante o desenvolvimento desse trabalho e mencionadas durante esse relatório.

1. Fundamentação Téorica

Nessa seção, iremos descrever conceitos que foram necessários para o entendimento do trabalho e técnicas que foram utilizadas para a compreensão e para começar a implementação desse projeto.

1. *Conceitos teóricos*

Primeiramente, IRC é a sigla pra *Internet Relay Chat*. Esse protocolo de aplicação atua na comunicação entre usuários da rede por meio de mensagens de texto, permitindo a um usuário entrar e criar um ou mais canais, onde ele poderá conversar com um ou mais usuários. Além disso, o protocolo também suporta o envio de mensagens diretas a outros usuários sem a necessidade de criar um canal. Esse protocolo foi muito utilizado no início dos anos 2000, porém, com o surgimento de outros aplicativos de conversa que surgiram utilizando outros protocolos, como WhatsApp e Telegram, seu uso tem diminuído.

Além disso, um cliente IRC é um software que atua na camada de aplicação da rede, a camada mais superior da pilha de protocolos TCP/IP. Portanto, é de responsabilidade do cliente fornecer com quem ele quer se comunicar, o conteúdo das mensagens, e a interpretação das mensagens que chegam, não sendo necessário à aplicação se preocupar com o trajeto dos pacotes na rede. Com isso, a aplicação deve montar as mensagens e disponibilizá-las em sockets, que são a porta de comunicação da camada de aplicação com o resto da pilha de protocolos, e também deve observar esse mesmo socket para observar a chegada de mensagens.

Com o objetivo de aprender sobre o funcionamento do protocolo IRC e suas mensagens, estudamos a RFC 2812, acessada em [1]. Nesse documento estão presentes todas as mensagens significantes para um cliente IRC, com seus significados, sua sintaxe, sinalizando todos os campos de cada mensagem e quais mensagens de erro o servidor poderia retornar após aquela mensagem ser enviada. Portanto, com esse documento, foi possível observar a sintaxe de mensagens do IRC, possíveis mensagens de erro e conceitos importantes.

Além disso, também foi importante descobrir como o servidor enviava mensagens como o PRIVMSG para os clientes IRC, para conseguirmos saber como decodificar esses dados. Para isso, estudamos [2], pois, como esse é um exemplo de um exercício da Universidade de Chicago voltado à criação de um servidor IRC, as interações explicadas mostraram como um servidor normalmente responde e encaminha certas mensagens. Dessa forma, foi possível descobrir a resposta do servidor a certas mensagens facilmente.

1. *Técnicas utilizadas*

Com o intuito de desenvolver a parte do cliente que efetivamente recebe e manda as mensagens na rede, utilizamos o módulo *socket* nativa do Python, que implementa uma interface de rede de baixo-nível, que é uma implementação da chamada de sistema do Unix utilizando a Orientação a Objetos do Python [3]. Com essa biblioteca, conseguimos enviar mensagens facilmente e receber quaisquer mensagens estivessem no buffer do *socket* quando necessário.

Para verificar se o nosso cliente IRC estava funcionando corretamente, foi necessário escolher algum servidor existente para nos conectarmos e testarmos o software. Já no começo do desenvolvimento, escolhemos nos conectar aos servidores DALnet, mais especificamente no servidor *serenity.fl.us.dal.net*, devido a uma maior proximidade geográfica em relação a outros servidores DALnet. Com isso, esse servidor foi selecionado para testarmos e analisarmos o comportamento durante o desenvolvimento do ChatWF.

Por fim, a técnica utilizada pelo grupo para conseguir trabalhar simultaneamente no código e ter um controle das mudanças feitas foi a utilização de um sistema de controle de versões, o *git*. Mais especificamente, foi criado um repositório no GitHub para que o código ficasse facilmente acessível a todos os integrantes do grupo. Logo, foi possível criar um ambiente de desenvolvimento em que é possível verificar quais mudanças cada pessoa fez separadamente.

1. Ambiente Experimental e Análise de Resultados

O projeto desenvolvido se encontra em um repositório do GitHub, visível no seguinte link: <https://github.com/fornari03/Cliente-IRC>

Para utilizar o ChatWF, execute o arquivo TelaAutenticacao.py, que está dentro da pasta src.

1. *Descrição do Cenário*

Para a confecção do projeto, foi utilizado o Visual Studio Code, da Microsoft, para o ambiente de desenvolvimento, utilizando-se a linguagem Python e a biblioteca socket padrão. Para a interface gráfica, utilizou-se o software PyQt Designer e a biblioteca PyQt5. O programa Wireshark, o *packet sniffer* escolhido pelo grupo, também foi amplamente usado durante o projeto, a fim de monitorar a transmissão dos pacotes IRC. Para o servidor, como mencionado na seção B. da Fundamentação Teórica, foram escolhidos majoritariamente os servidores de Internet Relay Chat do DALnet, para os testes e verificações no decorrer do trabalho. Além de todos esses, foram utilizados o git e o GitHub para versionamento e hospedagem do código.

1. *Transmissão de dados entre cliente e servidor*

É possível analisar a transmissão de dados entre o cliente desenvolvido e o servidor com o auxílio do Wireshark, ou outro *packet sniffer.* Abaixo, nas figuras 1 e 2, tem-se uma captura de um momento de funcionamento da aplicação, com dados tanto do cliente, quanto do servidor.

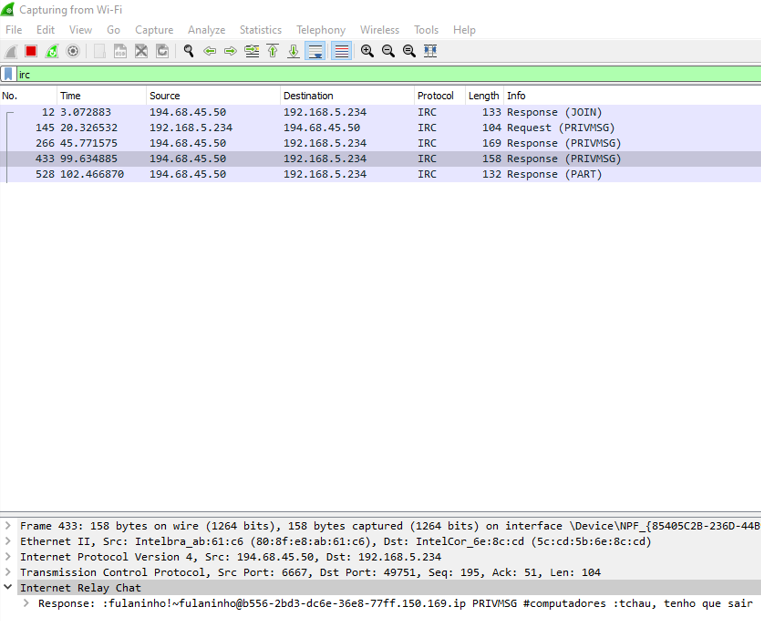


Fig. 1. Captura de pacotes durante o funcionamento do ChatWF utilizando o Wireshark

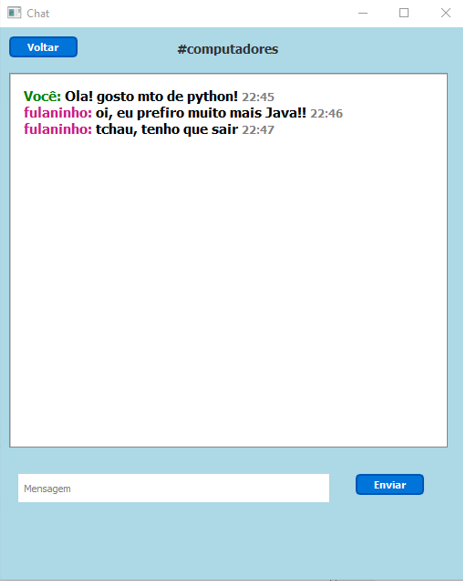


Fig. 2. ChatWF no mesmo momento da captura de pacotes da figura 1

Podemos ver um bom funcionamento do cliente desenvolvido, uma vez que os pacotes recebidos e enviados contêm as informações corretas. Primeiro, recebemos uma mensagem do servidor, visto que o pacote é uma resposta e origina do IP do servidor, dizendo que um novo usuário entrou no canal pela mensagem JOIN, sendo ele o “fulaninho”. Depois disso, o usuário enviou uma mensagem PRIVMSG expressando que gosta de Python, e logo recebe uma nova mensagem do servidor avisando que o “fulaninho” também enviou um PRIVMSG, falando que prefere Java. Em seguida, fulaninho envia outra PRIVMSG, se despedindo, e essa mensagem podemos observar seus detalhes no canto inferior esquerdo, vendo que a origem da mensagem é do usuário com nickname “fulaninho”, o tipo da mensagem é PRIVMSG, ela foi enviada ao canal #computadores, e a mensagem em si está após o caractere ‘:’, sendo que essa estrutura foi descrita em [1]. Por fim, o servidor envia uma última mensagem ao cliente nesse cenário informando que um usuário, nesse caso o “fulaninho”, saiu do canal, por meio da mensagem PART.

É importante lembrar que a camada de aplicação não é a responsável direta pelo envio dos pacotes. Este processo todo de envio e recibo de informações via rede depende de cada uma das camadas, são ela: camada de aplicação, camada de transmissão, camada de rede, camada de enlace e a camada física, de acordo com a abordagem da literatura científica do livro *Computer Networking: A Top-Down Approach*, 8ª edição, dos autores Jim Kurose e Keith Ross Pearson, de 2020 [4]. Abaixo na figura 3, podemos ver uma ilustração de como a mensagem, que começa na camada de aplicação, é transmitida numa arquitetura Cliente-Servidor até a camada de enlace.

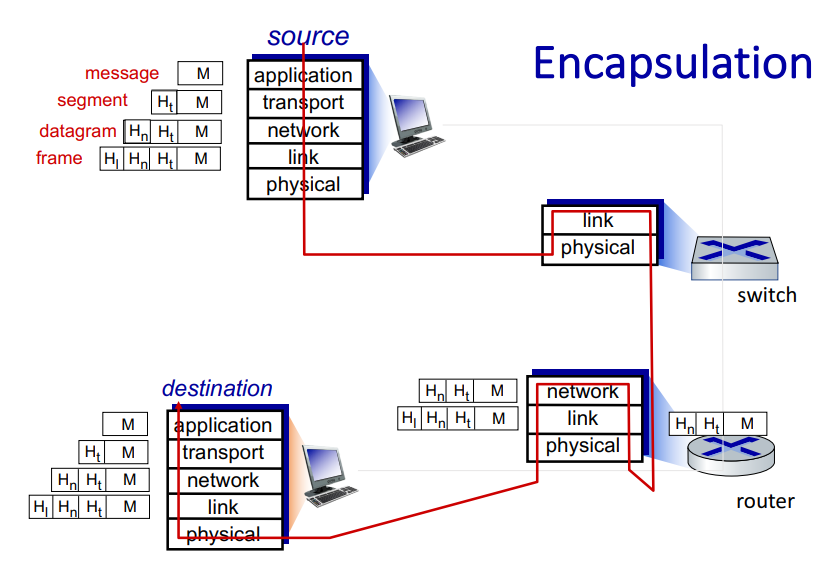


Fig. 3. Encapsulamento da mensagem até a camada de enlace, encontrada nos slides do capítulo 1 de [4]

1. *Análise de Resultados*

Pode-se dizer que o projeto, de certa forma, bem-sucedido, fazendo-se uma análise dos resultados esperados e dos resultados finais atingidos. Nesse contexto, esperava-se que o cliente fosse capaz de realizar as mais diversas atividades que um cliente IRC consegue fazer, e, portanto, vê-se que muitas delas foram implementadas com sucesso no ChatWF. Sempre com o auxílio da interface gráfica, o cliente consegue entrar em qualquer canal de um servidor, tanto por meio de pesquisa, quanto por meio da seleção na tabela de canais; o cliente consegue enviar e receber mensagens num canal, por intermédio do chat; consegue sair de um canal e de um servidor apenas clicando no botão de voltar nas respectivas telas; é possível também dar “refresh” na lista de canais, atualizando-a em tempo real. Dois pontos não tão positivos seriam que o cliente deve apenas conseguir se conectar a servidores do DALnet, visto que, quando o grupo estava implementando o ChatWF e utilizando dos servidores DALnet para testes, não foi percebido que as mensagens seriam diferentes em servidores de outros domínios. Portanto, não é garantido o funcionamento da aplicação para outros servidores que não sejam do DALnet, visto que se acreditou que haveria um padrão nas mensagens para todos. O outro ponto negativo é que não há como, por meio da interface, um usuário ter uma conversa privada com outro usuário em um servidor, ou seja, as conversas ocorrem sempre dentro de canais. Este impasse ocorreu devido a uma ausência de ideia do grupo em como isso poderia ser feito para a interface gráfica, pois somente com o PRIVMSG funcionando no cliente, era possível realizar conversas privadas. No entanto, como enunciado, o problema enfrentado foi como passaríamos isso para a interface, de maneira que não ficasse estranho ou com possíveis falhas na aplicação. Dessa forma, foi tomada a decisão de impossibilitar o usuário de mandar mensagens privadas. Além disso, não é extremamente incomum de ocorrer algum erro ou bug inesperado, o qual pode impedir o funcionamento do programa.

Vale ressaltar que uma das ideias iniciais do grupo era de fazer uma aplicação em que o usuário não tivesse que digitar nenhum comando, como \LIST ou \JOIN, vistos em algumas aplicações de cliente IRC que o grupo encontrou. A ideia do projeto seria proporcionar ao usuário uma experiência mais amigável e facilitada, o que foi atingido com êxito.

Ademais, foi gravado um vídeo, no formato de Pitch, em que é apresentado o ChatWF e é mostrado seu funcionamento, com alguns apontamentos e explicações da aplicação como um todo. O vídeo pode ser acessado a partir do link a seguir:

https://www.youtube.com/watch?v=ATqHcycGgKc

1. Conclusões

Portanto, esse trabalho consistiu na efetiva implementação de um cliente IRC. De acordo com os resultados obtidos, esse software é capaz de se conectar a servidores, entrar em canais e conversar com os usuários que também estiverem presentes nesse canal. Isso ocorre pela utilização de sockets que conectam esse programa da camada de aplicação ao resto da rede, pelo envio de requisições ao servidor e pela interpretação das mensagens enviadas por esse mesmo servidor.

1. Bibliografia

[1] KALT, C. **Internet Relay Chat: Client Protocol**. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2812>. Acesso em: 28 out. 2023.

**[**2] **Example IRC Communications — The UChicago χ-Projects**. Disponível em: <http://chi.cs.uchicago.edu/chirc/irc\_examples.html>. Acesso em: 31 out. 2023.

[3] **socket — Low-level networking interface**. Disponível em: <https://docs.python.org/3/library/socket.html#socket.socket.setblocking>. Acesso em: 28 out. 2023.

[4] KUROSE, J. F.; ROSS, K. W. **Computer networking : a top-down approach**. 8. ed. Hoboken: Pearson, 2021.