CHAT DE REDE LOCAL BASEADO NO PROTOCOLO TCP COM SERVIDOR EM MÁQUINA VIRTUAL LINUX

Dionathan da Rosa, João Pedro Oliveira Torres, Julio Samuel Rodrigues, Vinícios Matos Pellin e Adriano José Vogel

Faculdade Três de Maio (SETREM), Três de Maio, RS, Brasil.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de chat em rede local baseado no protocolo Transmission Control Protocol (TCP), implementado em Java com sockets. O objetivo foi projetar uma aplicação cliente-servidor, em que o servidor, executado em máquina virtual Linux (Ubuntu Server), gerencia conexões e encaminha mensagens, enquanto os clientes utilizam interface gráfica em Java Swing. A metodologia incluiu o estudo do TCP, a programação do sistema, a configuração da VM e testes em rede local. Entre as funcionalidades destacam-se o envio de mensagens públicas e privadas, a exibição de usuários conectados, a notificação de entrada e saída e o registro de data e hora. Os testes comprovaram a confiabilidade do TCP e a usabilidade da interface, demonstrando a viabilidade do projeto e reforçando conceitos de redes de computadores, programação distribuída e virtualização.

Palavras-chave: Cliente-Servidor. Comunicação. Java. Sockets. Transmissão de Dados.

1 INTRODUÇÃO

A comunicação em tempo real em redes locais é importante para a integração em ambientes acadêmicos e corporativos. O protocolo TCP, na camada de transporte do modelo TCP/IP, garante conexões confiáveis e ordenadas entre clientes e servidor. Este estudo foi motivado pela falta de soluções simples para troca de mensagens locais sem uso da internet e se justifica pelo valor acadêmico e prático, ao aplicar conceitos de redes de computadores, programação distribuída e virtualização no desenvolvimento de um chat em rede local com servidor Linux e clientes em Java Swing.

2 FUNDAMENTAÇÃO TEÓRICA

A partir desse tópico são descritos todos os conceitos que englobam o desenvolvimento do projeto do início ao final.

2.1 REDES DE COMPUTADORES

As redes de computadores permitem a interconexão de máquinas e dispositivos, viabilizando a troca de informações em diferentes camadas do modelo OSI. Na camada de aplicação, os protocolos garantem que os dados trafeguem de forma inteligível para o usuário final, possibilitando serviços como e-mails, navegação e chats (*TANENBAUM; WETHERALL*, 2011).

O protocolo TCP é amplamente utilizado no cotidiano em serviços como navegação web e troca de mensagens. Ele garante confiabilidade na entrega dos pacotes, mas apresenta maior sobrecarga em comparação a protocolos não orientados à conexão, como o UDP (*FOROUZAN*, 2012).

2.2 JAVA E SOCKETS

A linguagem Java se destaca por sua portabilidade e ampla aplicação em sistemas distribuídos. O uso de sockets permite a comunicação entre processos pela rede, tornando possível a implementação de aplicações cliente-servidor robustas (*DEITEL; DEITEL*, 2017).

Na construção de interfaces gráficas, o Java Swing oferece componentes reutilizáveis, como botões, caixas de texto e janelas. Essa biblioteca contribui para a usabilidade de aplicações, sem comprometer o desempenho da comunicação em rede (*ECKEL*, 2006).

2.3 SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos consistem em múltiplos computadores que funcionam de maneira integrada, compartilhando recursos e tarefas. Esse modelo permite escalabilidade, tolerância a falhas e eficiência no processamento, além de servir como base para serviços modernos de rede (*COULOURIS et al*, 2013).

2.4 SERVIDORES E VIRTUALIZAÇÃO

A arquitetura cliente-servidor estrutura-se na relação entre uma máquina central, que processa e gerencia os dados, e os clientes, que consomem os serviços disponibilizados. Esse modelo é aplicado em ambientes acadêmicos, empresariais e de simulação de redes (*STALLINGS*, 2012).

O uso de máquinas virtuais possibilita criar ambientes isolados para execução de sistemas, como servidores Linux em virtualizadores como VirtualBox. O Ubuntu Server, mesmo em linha de comando, garante estabilidade e suporte em simulações. A configuração de VMs com rede e pastas compartilhadas favorece a integração entre host e convidado, otimizando o desenvolvimento e os testes (*TANENBAUM; BOS*, 2015).

3 METODOLOGIA

O trabalho consistiu no desenvolvimento de um chat em rede local utilizando sockets TCP em Java, programado na plataforma IntelliJ IDEA. O servidor foi configurado em uma máquina virtual Linux (Ubuntu Server), enquanto os clientes utilizam interface gráfica em Swing[1].

A abordagem metodológica incluiu o estudo teórico de TCP, arquitetura cliente-servidor e programação distribuída, seguido da implementação prática do sistema. Foram aplicados procedimentos de prototipação incremental, permitindo desenvolver e testar funcionalidades em etapas, como envio de mensagens públicas e privadas, listagem de usuários conectados e registro de data e hora.

Para validar o sistema, realizou-se a instalação e configuração do Ubuntu Server 24.04 em máquina virtual, com ajustes de rede local e pastas compartilhadas com o host. Os testes experimentais ocorreram em três computadores na mesma rede, avaliando comunicação entre servidor e clientes, estabilidade do servidor e usabilidade da interface, assegurando o correto funcionamento do sistema em ambiente local.

4 APRESENTAÇÃO, ANÁLISE E DISCUSSÃO DOS RESULTADOS

A implementação do chat[2] demonstrou comunicação estável e confiável entre servidor e clientes. O servidor em Linux, mesmo apenas em linha de comando, gerenciou múltiplas conexões sem falhas, garantindo entrega ordenada das mensagens pelo TCP. O uso de máquina virtual trouxe isolamento do ambiente e

[1] Swing é um conjunto de ferramentas para criação de interfaces gráficas em Java. Mais informações em: https://docs.oracle.com/javase/8/docs/technotes/guides/swing/

[2] O código-fonte do projeto em Java está disponível em repositório público no GitHub: https://github.com/julioosr/chat-java

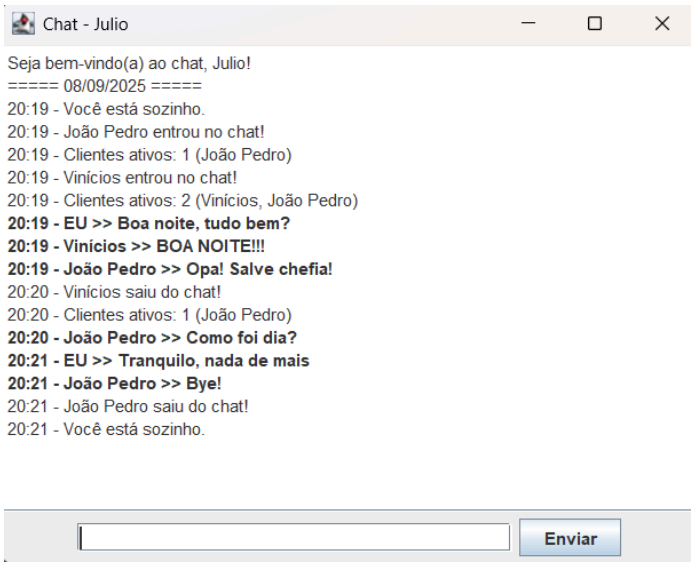
praticidade nos testes, evidenciando a viabilidade do recurso em simulações acadêmicas.

Figura 1 - Servidor em execução na VM Linux

Fonte: Do autor (2025).

A interface gráfica Swing permitiu envio de mensagens públicas e privadas, visualização de usuários conectados e notificações de entrada e saída. O registro de data e hora melhorou a organização e rastreabilidade da comunicação. Pequenas dificuldades ocorreram na configuração da rede e compatibilidade de versões do Java, sem comprometer os resultados.

Figura 2 - Interface gráfica do cliente em Java Swing



Fonte: Do autor (2025).

Os resultados comprovam a eficácia da abordagem utilizando o protocolo TCP em redes locais e a adequação do modelo cliente-servidor. O trabalho contribui academicamente ao demonstrar a integração prática de redes, programação distribuída e virtualização.

5 CONCLUSÃO

O trabalho atingiu seus objetivos, comprovando a viabilidade de um chat em rede local com servidor Linux e clientes em Java Swing. Funcionalidades como envio de mensagens, registro de data e hora e listagem de usuários conectados funcionaram corretamente, permitindo aplicar na prática conceitos de programação distribuída, redes de computadores e virtualização.

Como trabalhos futuros, recomenda-se avaliar a escalabilidade do servidor, testar a confiabilidade em redes com perdas de pacotes e implementar camadas de segurança na troca de mensagens, garantindo maior proteção e robustez do sistema.

6 REFERÊNCIAS

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos: conceitos e projeto.** 5. ed. Porto Alegre: Bookman, 2013.

DEITEL, H. M.; DEITEL, P. J. **Java: como programar.** 10. ed. São Paulo: Pearson, 2017.

ECKEL, B. **Thinking in Java.** 4. ed. New Jersey: Prentice Hall, 2006.

FOROUZAN, B. A. **Comunicação de dados e redes de computadores.** 4. ed. Porto Alegre: McGraw-Hill, 2012.

STALLINGS, W. **Arquitetura e organização de computadores.** 8. ed. São Paulo: Pearson, 2012.

TANENBAUM, A. S.; BOS, H. **Sistemas operacionais modernos.** 4. ed. São Paulo: Pearson, 2015.

TANENBAUM, A. S.; WETHERALL, D. **Redes de computadores.** 5. ed. Rio de Janeiro: Elsevier, 2011.