Desenvolvimento de um Aplicativo de Mensagens em Arquitetura Peer-to-Peer: Uma Abordagem Descentralizada

Ana J. O. Silva[1], **Ícaro Machado**[1], **Caua V. F. Dalmeida**[1], **Pedro H. F. Silva**[1]

[1]Instituto Federal da Paraíba (IFPB) – Tecnólogo em Redes de Computadores

Resumo. *Esta pesquisa visa o desenvolvimento de um aplicativo Peer-to-Peer (P2P) para facilitar a comunicação direta entre usuários sem a necessidade de um servidor central. O objetivo é criar uma plataforma eficiente, segura e escalável, com suporte para troca de mensagens em tempo real. A arquitetura P2P permite que os usuários atuem tanto como clientes quanto como servidores, melhorando a distribuição de recursos e otimizando a utilização de banda. O desenvolvimento foi realizado em Python, utilizando bibliotecas como* `socket`, `threading` *e* `hashlib` *para comunicação, gerenciamento de conexões e criptografia, respectivamente. Após a implementação, o aplicativo foi submetido a testes de conectividade, desempenho e segurança, com resultados que demonstraram a viabilidade da solução proposta. Conclui-se que a arquitetura P2P oferece uma alternativa promissora para sistemas de mensagens descentralizados, com potencial para aplicações em redes locais e corporativas. Futuros trabalhos podem explorar a escalabilidade do sistema e a implementação de interfaces gráficas para melhorar a experiência do usuário.*

Palavras-chave: *aplicativos mensageiros, desenvolvimento de software, redes P2P.*

Abstract. *This research aims to develop a Peer-to-Peer (P2P) application to enable direct communication between users without the need for a central server. The focus of the development is to create an efficient, secure, and scalable platform that supports real-time data exchange between devices. The P2P architecture allows users to act as both clients and servers, improving resource distribution and optimizing bandwidth usage. After implementation, the application will undergo a series of tests to evaluate its performance, security, and stability. At the end of the process, the test results will be analyzed to validate the application's effectiveness and identify areas for future improvements and optimizations, with the goal of enhancing the user experience and the robustness of the P2P platform.*

1. Introdução

Os sistemas de mensagens instantâneas, como WhatsApp, Telegram e Discord, tornaram-se pilares da comunicação digital moderna. Essas plataformas, no entanto, são baseadas em uma arquitetura cliente-servidor, na qual os usuários dependem de servidores centralizados para trocar mensagens. Embora esse modelo tenha facilitado a adoção em larga escala, ele apresenta desafios significativos, como vulnerabilidades à centralização, riscos de falhas sistêmicas e questões de privacidade e segurança. A dependência de um servidor central não apenas expõe os usuários a possíveis interrupções do serviço, mas também cria um ponto único de falha que pode ser explorado por ataques cibernéticos.

A arquitetura peer-to-peer (P2P) surge como uma alternativa viável para mitigar essas limitações. Conforme destacado por COELHO (2008), ”o termo P2P refere-se a uma classe de sistemas e aplicações que empregam recursos distribuídos para executar uma função de forma descentralizada”. Nesse modelo, cada dispositivo atua simultaneamente como cliente e servidor, permitindo a comunicação direta entre os usuários sem a necessidade de um servidor central. Essa abordagem descentralizada reduz a dependência de um ponto único de falha, melhora a resiliência da rede e proporciona maior autonomia aos usuários. Além disso, a arquitetura P2P pode oferecer maior privacidade, já que as mensagens não precisam passar por um servidor centralizado.

Apesar das vantagens, a implementação de sistemas P2P para mensagens instantâneas ainda enfrenta desafios técnicos, como o gerenciamento eficiente de conexões, a prevenção de loops e a garantia de segurança nas comunicações. Trabalhos anteriores, como o mensageiro P2P desenvolvido por KOMO (2018), demonstraram a viabilidade dessa abordagem, mas poucos estudos exploraram soluções práticas e escaláveis para redes locais ou corporativas. Essa lacuna no conhecimento motivou o desenvolvimento deste projeto, que busca propor uma solução P2P para mensagens instantâneas, com foco em resiliência, segurança e facilidade de implementação.

O objetivo principal deste estudo é desenvolver e avaliar um aplicativo de mensagens P2P que supere as limitações dos sistemas centralizados tradicionais. A hipótese central é que a arquitetura P2P pode oferecer um ambiente mais seguro, resiliente e eficiente para a troca de mensagens, especialmente em contextos onde a privacidade e a autonomia são prioritárias. Para testar essa hipótese, o projeto inclui a implementação de funcionalidades essenciais, como autenticação de usuários, criptografia de dados e gerenciamento dinâmico de conexões.

A relevância desta pesquisa reside em sua contribuição para o avanço de soluções de comunicação descentralizadas. Ao abordar os desafios técnicos e propor soluções inovadoras, este trabalho serve como um exemplo prático para futuras aplicações que busquem explorar os benefícios da arquitetura P2P. Além disso, a comparação entre o modelo P2P e os sistemas centralizados tradicionais evidencia as vantagens da descentralização, inspirando o desenvolvimento de sistemas mais eficientes, seguros e adaptáveis às necessidades dos usuários.

2. Fundamentação Teórica

A Internet, como conhecemos hoje, é um pilar essencial da infraestrutura global de comunicação. Grande parte de sua estrutura é baseada no modelo cliente-servidor, no qual os usuários dependem de servidores centralizados para acessar recursos e serviços. No entanto, esse modelo apresenta limitações, como a dependência de um núcleo centralizado e questões relacionadas à privacidade e à escalabilidade. Como destacado por COELHO (2001), ”a Internet surgiu como meio de se compartilhar livre e incondicionalmente o conhecimento e os recursos disponibilizados em uma rede de dados, mas sempre se baseou na estrutura cliente-servidor, o que torna os usuários dependentes de um núcleo que detenha e gerencie esse conhecimento ou recurso”.

Para superar essas limitações, redes peer-to-peer (P2P) foram desenvolvidas como uma alternativa descentralizada. Nessas redes, cada host funciona tanto como cliente quanto como servidor, eliminando a necessidade de um servidor central. Conforme CO-

ELHO (2008) explica, ”o termo P2P refere-se a uma classe de sistemas e aplicações que empregam recursos distribuídos para executar uma função de forma descentralizada. Muito se tem falado atualmente deste modelo de interligação de redes de computadores, dando a ideia para muitos que este modelo é uma tecnologia recente. No entanto, a área de Sistemas Distribuídos já utiliza esse modelo de pontos (também chamados nós) altamente acoplados para compartilhamento de recursos computacionais, como forma de aumentar o desempenho na resolução de problemas complexos”.

No contexto de aplicativos de mensagens, as redes P2P oferecem vantagens significativas, como maior privacidade (já que as mensagens não passam por um servidor central) e resiliência (já que a rede não depende de um único ponto de falha). No entanto, também apresentam desafios, como o gerenciamento eficiente de conexões, a prevenção de loops e a garantia de segurança nas comunicações. Um exemplo relevante é o mensageiro P2P desenvolvido por KOMO (2018), que demonstrou a viabilidade de aplicativos descentralizados para comunicação em tempo real, servindo como inspiração para o desenvolvimento deste projeto.

A implementação de redes P2P depende de tecnologias e técnicas fundamentais, como sockets para comunicação em rede, threading para gerenciamento de conexões simultâneas e criptografia para garantir a segurança dos dados. Sockets são a base da comunicação em sistemas distribuídos, permitindo a troca de dados entre dispositivos em uma rede. Threads, por sua vez, são essenciais para lidar com múltiplas conexões de forma concorrente, garantindo que o sistema permaneça responsivo. Por fim, a criptografia desempenha um papel crucial na proteção de informações sensíveis, como credenciais de usuários, garantindo que apenas partes autorizadas tenham acesso aos dados.

3. Desenvolvimento

A linguagem de programação para desenvolver a aplicação escolhida foi o Python, por ser uma linguagem de alto nível que abstrai detalhes do gerenciamento de memória, execução e interação com o hardware, tornando a programação mais intuitiva e próxima da linguagem humana. Além disso, seu interpretador segue a filosofia batteries included, oferecendo uma ampla integração nativa de bibliotecas, o que facilita o desenvolvimento sem a necessidade de configurações complexas. Como defendido por autores como BORGES, escritor de Python para desenvolvedores.

Foram utilizados os seguintes recursos: as bibliotecas socket para comunicação em rede; threading para gerenciamento de conexões simultâneas; json para armazenamento de credenciais; os para interação com o sistema operacional; logging para registro de atividades e hashlib para criptografia das senhas dos usuários.

A estrutura do aplicativo é dividida em dois módulos principais: Cliente e Servidor. O módulo Cliente estabelece conexões e envia mensagens para outras instâncias, na mesma rede ou máquina. Já o módulo Servidor escuta por conexões recebidas e, ao detectar uma nova conexão, a gerencia em uma thread separada, permitindo múltiplas conexões simultâneas. Cada thread é responsável por receber mensagens de outros usuários e exibi-las na interface. A Figura 1 ilustra a abstração dos módulos.

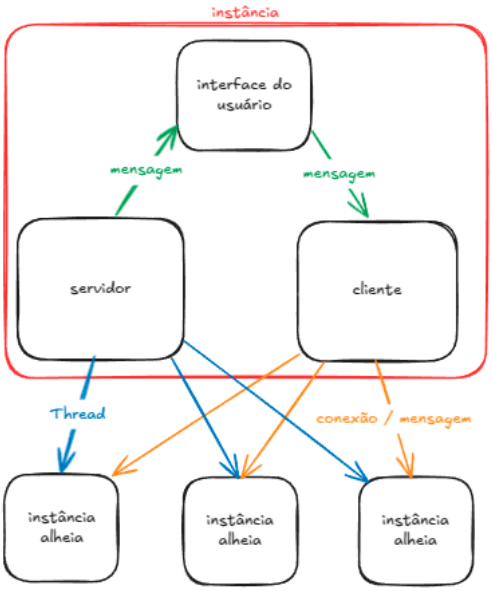

Figura 1. Abstração dos módulos do aplcativo.

A interface de usuário do aplicativo é o terminal onde o arquivo é executado. Na primeira instância, o programa executa um cadastro de conta simples, com nome de usuário e senha. Depois disso, a cada instância do programa, o usuário é autenticado e escolhe uma porta fixa de escuta para o servidor. Depois de devidamente iniciado, o terminal funciona como um bate-papo em grupo de um aplicativo funcional: O usuário escreve uma mensagem, pressiona enter e a mensagem é enviada para todos os peers conectados.

O programa também conta com um sistema de comandos, para a execução de tarefas específicas. Por exemplo, o comando /connect estabelece uma conexão com o par de endereço IP e porta especificado. O comando /peers mostra a lista de peers conectados, Etc...

O algoritmo de estabelecimento de conexão funciona da seguinte forma: quando uma instância A do programa se conecta unilateralmente a uma instância B, a instância A compartilha sua base de dados de peers já conectados. A instância B, ao receber essa base de dados, varre a lista de endereços e estabelece conexões com cada um deles, incluindo a instância A. Dessa forma, cria-se uma conexão bidirecional entre as instâncias, permitindo o envio de mensagens em ambas as direções. Para evitar loops e conexões redundantes, o algoritmo inclui um sistema de verificação que impede que uma instância se conecte a si mesma ou duplique conexões com peers já conectados. A Figura 2 detalha o funcionamento do algoritmo.

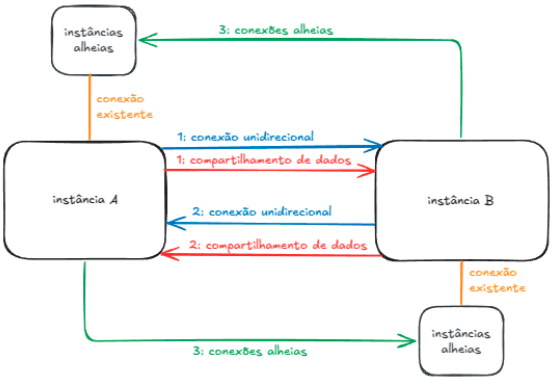

Figura 2. Funcionamento do algoritmo de conexão.

O desenvolvimento do aplicativo começou com uma implementação básica de comunicação via sockets. Após várias refatorações e testes de diferentes algoritmos de conexão, a versão final do código foi definida, e novas funcionalidades foram incrementadas sobre essa base.

4. Resultados

Para a fase de testes, foram utilizadas quatro instâncias do aplicativo: A, B, C e D. Inicialmente, A conectou-se a B, e C conectou-se a D, formando dois pares isolados (Figura 4). Em seguida, uma conexão foi estabelecida entre A e C, resultando na integração das conexões A-B e C-D em um único grupo de bate-papo (Figura 5). A Figura 3 ilustra os resultados obtidos nos testes, demonstrando a eficácia do algoritmo de conexão.

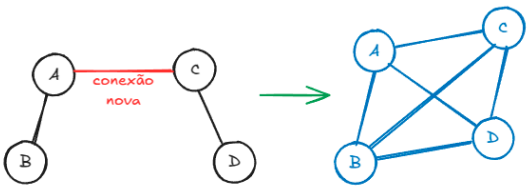

Figura 3. Resultados dos testes em diagrama.

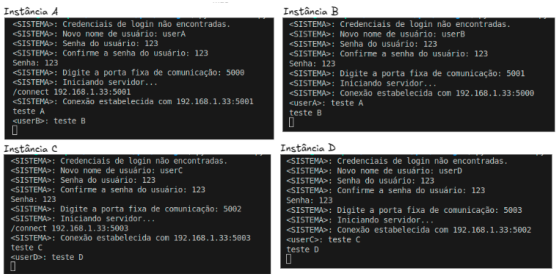

Figura 4. Resultados dos testes.

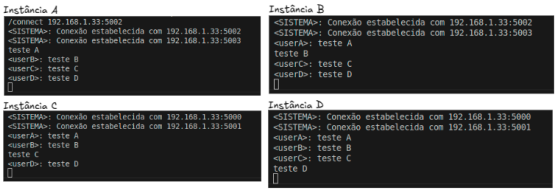

Figura 5. Resultados dos testes.

Estes mesmos testes foram realizados em um ambiente controlado, com 4 máquinas diferentes, cada uma executando uma instância diferente do programa. Os resultados obtidos foram os mesmos.

5. Discussão

Os resultados obtidos com o aplicativo de mensagens P2P demonstraram a viabilidade e a eficácia da solução proposta. A estrutura descentralizada permitiu a conexão direta entre os dispositivos, garantindo um fluxo de comunicação dinâmico e eficiente. O algoritmo de conexão foi capaz de estabelecer ligações entre os pares sem gerar loops ou redundâncias, garantindo que a rede fosse montada de forma organizada e otimizada.

Durante os testes, foi possível observar que o compartilhamento de listas de peers entre as instâncias facilitou a expansão da rede, permitindo que novos nós fossem incorporados sem a necessidade de um servidor centralizado. Esse mecanismo mostrou-se eficaz na construção de uma rede robusta e interconectada, essencial para um sistema de comunicação descentralizado.

A segurança foi um dos pontos fundamentais analisados. O uso de criptografia garantiu a proteção das mensagens durante o trânsito, reduzindo o risco de interceptação por terceiros. No entanto, futuras melhorias podem ser implementadas para fortalecer ainda mais a segurança do sistema, como a adição de mecanismos de autenticação multifator e um modelo aprimorado de gerenciamento de chaves criptográficas.

Em relação ao desempenho, os testes mostraram que a aplicação conseguiu manter um bom nível de responsividade mesmo com múltiplas conexões simultâneas. O uso de threads para o gerenciamento de conexões permitiu a distribuição eficiente da carga de processamento, evitando gargalos que poderiam comprometer a usabilidade do sistema.

Entretanto, alguns desafios foram identificados. O sistema se mostrou escalável com os testes feitos até então. Contudo, não se sabe até quantas conexões o aplicativo é capaz de gerenciar. Além disso, uma interface gráfica intuitiva pode ser desenvolvida para aumentar a qualidade de serviço para o usuário final. Todavia, os resultados positivos dos testes apontam o potencial retorno de aplicativos análogos, reforçando a importância de soluções descentralizadas no cenário da comunicação digital, especialmente em um contexto onde privacidade, segurança e independência são cada vez mais valorizadas. Futuras pesquisas e desenvolvimentos podem explorar novos protocolos de comunicação P2P, otimização de algoritmos de conexão e aprimoramento dos mecanismos de proteção de dados, visando consolidar ainda mais essa abordagem como uma alternativa viável e segura para mensageiros instantâneos.

6. Conclusão

Esta pesquisa demonstrou a viabilidade de um aplicativo de mensagens baseado em arquitetura P2P, oferecendo uma alternativa descentralizada e segura para a comunicação digital. Os resultados dos testes mostraram que o sistema é capaz de gerenciar múltiplas conexões simultaneamente, com bom desempenho e segurança. A implementação de funcionalidades como autenticação de usuários, criptografia de dados e gerenciamento dinâmico de conexões contribuiu para a robustez e a eficiência do sistema.

A principal contribuição deste trabalho reside na demonstração prática dos benefícios da arquitetura P2P para sistemas de mensagens, especialmente em contextos onde a privacidade e a autonomia são prioritárias. Futuras pesquisas podem explorar a escalabilidade do sistema, a implementação de interfaces gráficas e o aprimoramento dos mecanismos de segurança, consolidando ainda mais essa abordagem como uma alternativa viável e segura para mensageiros instantâneos.

Referências

[1] COELHO, Edinilson Machado. *Redes P2P: análise de aplicativos P2P*. 2008. Monografia (Especialização em Gerência de Redes de Computadores) — Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

[2] KOMO, Andrea E.; ARAKAKI, Bruno O.; SIMPLICIO JR., Marcos A.; LEVY, Mayer R. *Aplicativo de troca de mensagens instantâneas utilizando comunicação P2P*. São Paulo: Escola Politécnica, Universidade de São Paulo, 2018.

[3] BORGES, Luiz Eduardo. *Python para Desenvolvedores*. 3. ed. São Paulo: Novatec, 2014.

Apêndices

Apêndice A

Código fonte: `https://github.com/icaruuuuuu/ptc`