Chat Peer-to-Peer

João Marcelo Rossi Matheus Freire Rafael Torga

Fevereiro, 2025

Resumo

Este trabalho visa implementar uma aplicação simplificada de um chat Peer-to-Peer, visando demonstrar a compreensão dos conceitos abordados na disciplina de Sistemas Distribuídos.

Palavras-chaves: Peer to Peer, Chat, Sistemas Distribuidos

Introdução

Redes Peer-to-Peer (P2P) são um modelo de comunicação onde cada dispositivo conectado pode atuar, ao mesmo tempo, como cliente e servidor. Ao contrário do modelo tradicional cliente-servidor, que depende de um servidor central para gerenciar a troca de informações, as redes P2P permitem que os dispositivos se comuniquem diretamente entre si. Essa abordagem é bastante comum em aplicações que exigem o compartilhamento eficiente de recursos, como na transferência de arquivos, no streaming descentralizado e, especificamente, em sistemas de chat baseados nesse modelo (SCHOLLMEIER, 2001).

O presente trabalho tem como objetivo desenvolver e analisar um sistema de chat baseado em redes P2P local, explorando tanto suas potencialidades quanto suas limitações. A proposta é demonstrar a viabilidade deste modelo de comunicação, identificando os desafios técnicos envolvidos e propondo soluções que contribuam para melhorar o desempenho do sistema.

1 Objetivo

Desenvolver uma aplicação de chat Peer-to-Peer que permite a comunicação de 2 pares entre si. A implementação utilizará uma espécie de Superpeer que irá armazenar a lista de pares disponíveis na rede P2P.

2 Contexto

Implementar um sistema de chat utilizando a arquitetura P2P traz várias vantagens. Uma delas é a escalabilidade: cada novo usuário que se conecta à rede adiciona recursos, tornando o sistema mais robusto e aumentando sua capacidade de processamento e comunicação. Além disso, como a infraestrutura é formada pelos próprios dispositivos dos usuários, os custos operacionais são significativamente reduzidos, dispensando a necessidade de servidores dedicados. Outro aspecto positivo é a descentralização, que torna o sistema mais tolerante a falhas. Se um dispositivo se desconectar, a rede continua funcionando sem comprometer a comunicação entre os demais nós (STOICA et al., 2001).

Por outro lado, essa arquitetura também apresenta desafios importantes. A complexidade na implementação é um dos principais obstáculos, pois é necessário desenvolver mecanismos eficientes para descobrir os pares, gerenciar o roteamento das mensagens e sincronizar os dados entre os nós. Questões de segurança, como a proteção contra ataques maliciosos e a garantia da privacidade dos usuários, também exigem cuidados especiais. Além disso, o gerenciamento dos recursos pode ser mais difícil sem um servidor central para coordenar a distribuição de banda e de poder de processamento.

Esse modelo é amplamente utilizado em diversas aplicações, sendo um dos exemplos mais conhecidos o protocolo BitTorrent, que permite a distribuição eficiente de arquivos entre múltiplos usuários sem a necessidade de um servidor centralizado. Outra aplicação relevante do modelo P2P são os chats descentralizados, como o Tox, que possibilita comunicação segura entre usuários sem depender de servidores intermediários (ROWSTRON; DRUSCHEL, 2001).

No caso específico de chats P2P, a descentralização oferece vantagens como resistência à censura, já que não há um servidor central que possa ser bloqueado ou monitorado, além de maior privacidade, pois as mensagens trafegam diretamente entre os usuários sem

necessidade de armazenamento em servidores terceiros. Isso torna essa abordagem ideal para comunicações seguras, ainda mais em cenários de repressão.

3 Metodologia

A metodologia adotada para o desenvolvimento do sistema de chat P2P envolveu a utilização da linguagem Python, com as bibliotecas socket, threading e Tkinter. Inicialmente, foi implementada a comunicação em rede utilizando sockets TCP, permitindo que cada nó atuasse tanto como servidor quanto como cliente. Para suportar múltiplas conexões simultâneas, empregou-se a técnica de multithreading, onde cada conexão é gerenciada em uma thread individual, garantindo o envio e o recebimento de mensagens de forma concorrente.

Além disso, foi introduzido um componente denominado super peer, responsável pelo registro e pela descoberta dos nós participantes. Os peers se registram no tracker enviando mensagens padronizadas, o que facilita a manutenção de uma lista atualizada dos dispositivos disponíveis para comunicação, sem centralizar o conteúdo das mensagens trocadas. Para retornar e buscar mensagens, os pares e o super peer utilizam a lógica ensinada em sala, Ping Pong, em que enviam um sinal para o super peer "perguntando"os servidores disponíveis e o super peer retorna.

A interface gráfica foi desenvolvida com a biblioteca Tkinter, proporcionando uma experiência amigável ao usuário por meio de componentes como caixas de texto, campos de entrada e botões para envio de mensagens e gerenciamento de conexões. Essa camada separada da lógica de rede permitiu uma melhor organização do código e facilitou futuras expansões.

Em suma, a abordagem metodológica combinou técnicas de programação de rede, concorrência e desenvolvimento de interfaces, resultando em um sistema de chat P2P que explora as vantagens da descentralização, mantendo, ao mesmo tempo, um gerenciamento centralizado parcial por meio do tracker para otimizar a descoberta e a conexão entre os nós.

4 Como Implementar

Não existem grandes requisitos técnicos, fora uma máquina com acesso a internet e um ambiente de python que permita a adição de novas bibliotecas.

1. Configuração Inicial

   Crie um socket TCP para comunicação

   Use threading para operações simultâneas

   Implemente a interface gráfica com Tkinter

2. Super Peer Setup

   Garanta que o Super Peer rode em um IP/porta específicos

   Implemente funcionalidade básica (registro/listagem de peers)

3. Autodescoberta de IP

   implemente uma classe para registrar o IP local do usuário

Isso permite conexões diretas entre peers

4. Registro no Super Peer

   Crie uma função que irá fazer essa ligação entre os peers e o super peer.

   Ao iniciar, cada peer envia REGISTER [IP] [PORT] [USERNAME] ao Super Peer

5. Servidor de Escuta

   Cada peer deve iniciar um servidor (listener) em sua porta

   Aceita conexões entrantes em threads separadas

6. Atualização de Peers Disponíveis

   Botão "Atualizar Peers"chama uma função que bate no Super peer

   Consulta o Super Peer e atualiza a Listbox com peers ativos

7. Conexão com Peers

   Ao selecionar um peer da lista, envia uma conexão para o par

   Valida auto-conexão e conexões duplicadas antes de conectar

8. Gerenciamento de Conexões

   uma função deve gerenciar mensagens recebidas

   Mantém lista de sockets conectados (peers)

9. Envio de Mensagens

   uma função deve enviar texto para todos os peers conectados

   Atualiza a interface local simultaneamente

10. Tratamento de Erros

    Implemente timeouts e reconexões

    Use try/except para falhas de conexão

    Mostra alerts com messagebox para erros críticos

5 Resultados

Abaixo veremos o resultado da implementação com 2 máquinas diferentes testando a aplicação de chat. Observe que um peer possui o usuário chamado "opa"e o outro possui o usuário chamado "pc2".

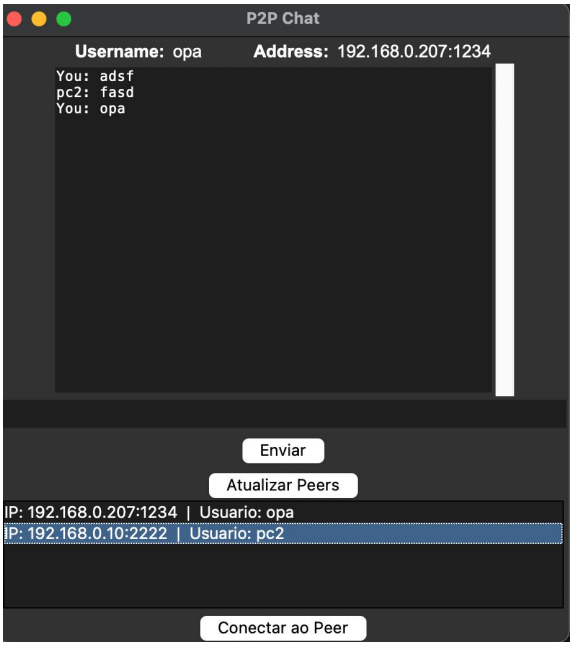

Figura 1 – Exemplo de conexão do opa com o pc2

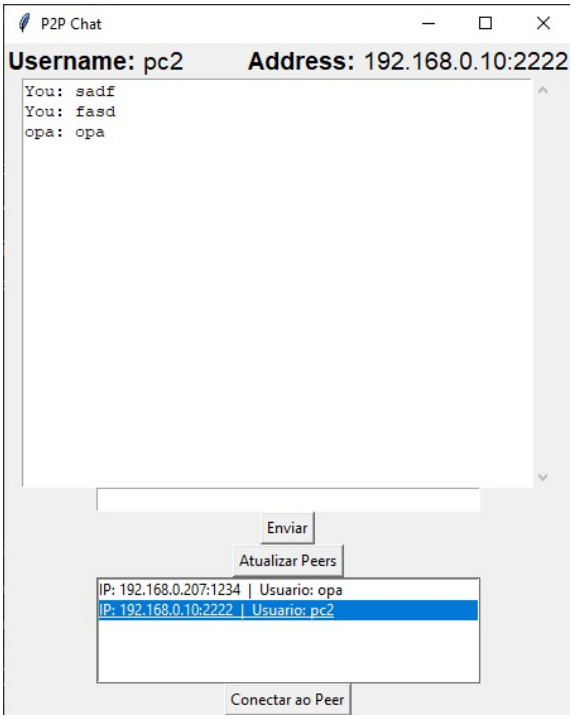

Figura 2 – Exemplo de conexão do pc2 com o opa

Acima vemos a interface gráfica da aplicação funcionando para os dois peers, perceba que a funcionalidade de mensagens, atualização de peers e conexão tiveram que ser utilizadas para realizar o teste.

Abaixo vemos algumas notificações que podem ser encontradas pelo usuário.

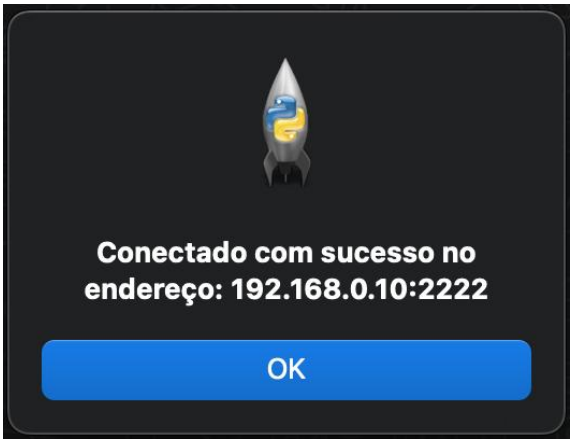

Figura 3 – Conectado com sucesso

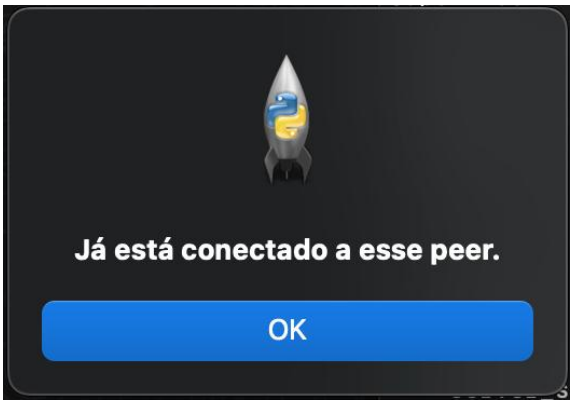

Figura 4 – Erro - tentativa de conexão dupla

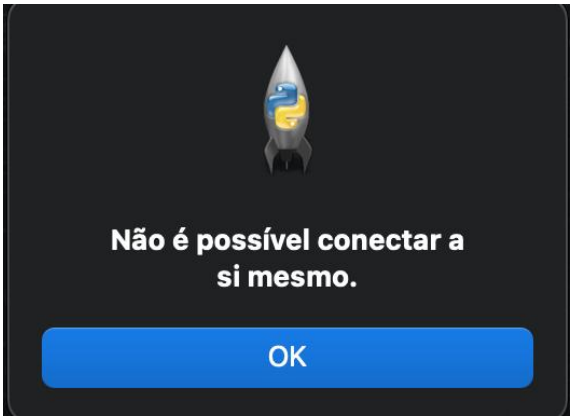

Figura 5 – Erro - tentativa de auto conexão

6 Desafios

Um dos principais desafios está na manutenção dos peers e no gerenciamento das conexões. Como os nós podem entrar e sair da rede de forma dinâmica, é necessário imple-

mentar mecanismos eficientes para manter uma lista de peers ativos, detectar desconexões e atualizar as conexões conforme necessário.

Além desses aspectos, um grande desafio para a implementação de um chat P2P é a conectividade entre redes distintas. Em muitos casos, os usuários estão atrás de roteadores NAT (Network Address Translation) ou firewalls, que dificultam a comunicação direta entre dispositivos. Para superar essa limitação, são necessárias técnicas como perfuramento de NAT (NAT Traversal), STUN (Session Traversal Utilities for NAT) e TURN (Traversal Using Relays around NAT), que permitem estabelecer conexões entre redes diferentes sem depender de um servidor intermediário.

Um dos exemplos normalmente utilizados para resolver o problema citado para implementar uma estrutura entre redes distintas foi a utilização de tunelamento, isso é, a técnica usada para criar um caminho seguro e direto entre dois dispositivos em redes diferentes quando há obstáculos como NATs ou firewalls que dificultam a comunicação direta. No entanto, ele pode introduzir dependências externas que quebram a característica de descentralização e por isso não funcionaria na implementação desejada.

7 Próximos Passos

Os próximos passos para o projeto envolvem a expansão e o aprimoramento do sistema de chat P2P, visando sua aplicação em diferentes tipos de redes e ambientes. Inicialmente, pretende-se adaptar a solução para operar eficientemente não apenas em redes locais, mas também em redes amplas (WAN) e em conexões móveis, o que exigirá a implementação de técnicas para atravessar NATs e firewalls. Essa adaptação é essencial para garantir a conectividade dos peers mesmo em ambientes com restrições de comunicação.

Além disso, será investigada a integração de mecanismos avançados de segurança, como criptografia de ponta a ponta e autenticação robusta, a fim de proteger as trocas de mensagens e garantir a privacidade dos usuários. A otimização do tracker (super peer) também está entre as prioridades, buscando melhorar o gerenciamento e a descoberta de nós ativos, especialmente em cenários com alta dinâmica (churn).

Outros desenvolvimentos previstos incluem a criação de uma interface multiplataforma, que possa ser utilizada tanto em desktops quanto em dispositivos móveis, ampliando o alcance do sistema. Paralelamente, serão conduzidos testes em ambientes reais para avaliar o desempenho e a escalabilidade do chat em situações de alta latência e tráfego intenso. Essas etapas fornecerão subsídios para futuras pesquisas na área de redes distribuídas, contribuindo para o aprimoramento de sistemas P2P e para o desenvolvimento de soluções de comunicação mais robustas e seguras.

8 Conclusão (Maiores aprendizados e o que mais gostou de fazer)

O desenvolvimento de um sistema de chat Peer-to-Peer (P2P) proporcionou uma experiência valiosa ao longo do processo, permitindo-nos enfrentar diversos desafios técnicos e compreender a complexidade das redes descentralizadas. Durante o trabalho, aprendemos que um dos maiores obstáculos na criação de uma rede P2P eficiente é garantir a comunicação estável e a manutenção dinâmica das conexões entre os peers.

A implementação de um sistema P2P dentro de uma rede local exigiu que desenvolvêssemos soluções para gerenciar a descoberta de peers, a detecção de desconexões e a

entrega confiável de mensagens. Essas questões foram abordadas com sucesso utilizando o protocolo de sockets em Python, o que permitiu uma troca de mensagens eficiente dentro do ambiente da mesma rede.

O que mais nos agradou neste projeto foi a oportunidade de aplicar os conceitos aprendidos na prática, especialmente no que diz respeito à comunicação entre dispositivos e a criação de uma rede descentralizada de pares.

Por fim, conseguimos aplicar técnicas de programação e redes para criar uma solução funcional dentro de uma rede local. O aprendizado obtido ao lidar com questões como gerenciamento de conexões e confiabilidade das mensagens foi essencial para nosso entendimento das redes descentralizadas,

Referências

ROWSTRON, A.; DRUSCHEL, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IEEE. *Middleware '01: Proceedings of the IFIP/IEEE International Conference on Distributed Computing Systems*. [S.l.], 2001. p. 329–350. Citado na página 2.

SCHOLLMEIER, R. A definition of peer-to-peer networking for the classification of distributed applications. In: IEEE. *Proceedings of the First International Conference on Peer-to-Peer Computing*. [S.l.], 2001. p. 101–102. Citado na página 2.

STOICA, I. et al. Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM. *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*. [S.l.], 2001. p. 149–160. Citado na página 2.