Rede P2P para transmissão de arquivos

Débora A. Caetano

Decisões de implementação

Definições gerais :

A aplicação utiliza um modelo P2P com apoio de um *tracker* centralizado, responsável por coordenar a rede. O tracker não armazena arquivos, sua função é registrar peers, manter informações atualizadas sobre quais arquivos cada peer possui e indicar quais deles estão ativos. Cada peer atua simultaneamente como cliente (fazendo solicitações ao tracker e a outros peers) e como servidor (enviando partes de arquivos quando solicitado). A comunicação ocorre por meio de sockets TCP, com uso de *heartbeat* para detecção de peers inativos e múltiplas threads para permitir buscas, downloads e atendimentos simultâneos.

1. Comandos presentes no projeto P2P:

Comandos que o peer pode solicitar (cliente)

- **exit**: remove o peer da rede e encerra sua conexão com o tracker.
- **list**: solicita ao tracker a lista de peers atualmente ativos.
- **search <nome_arquivo>**: procura quais peers possuem o arquivo informado.
- **download <nome_arquivo>**: inicia o processo de download do arquivo a partir dos peers que o possuem.

Comandos que o peer pode receber (servidor)

- **GET <arquivo> <offset> <size>**: envia ao solicitante o chunk correspondente do arquivo, começando em offset e com tamanho size.

Comandos que o tracker pode receber

- **register**: registra um novo peer na rede.
- **search**: retorna a lista de peers que possuem determinado arquivo.
- **unregister**: remove um peer da lista de ativos.
- **list**: envia a lista atual de peers ativos.
- **heartbeat**: atualiza o tempo da última atividade do peer.
- **update**: atualiza a lista de arquivos anunciados por um peer já registrado.

2. Fluxo geral do sistema:

1. **Inicialização do Tracker**
   O tracker é iniciado e começa a escutar conexões de peers. Ele mantém duas estruturas principais:

   - uma lista/dicionário de peers ativos,
   - e um registro de quando cada peer foi visto pela última vez.
     Paralelamente, uma thread de limpeza verifica periodicamente peers inativos.

2. **Inicialização de um Peer**
   Quando um peer é executado, ele abre seu servidor interno (para enviar arquivos a outros peers) e, em seguida, se conecta ao tracker.

3. **Registro no Tracker**
   O peer envia ao tracker o comando **REGISTER**, informando seu IP, porta e a lista de arquivos que possui.
   O tracker armazena essas informações.

4. **Envio periódico de Heartbeats**
   O peer inicia uma thread que, a cada intervalo definido, envia **HEARTBEAT** ao tracker. O tracker atualiza o horário da última atividade do peer.

5. **Busca de Arquivos**
   Quando o peer solicita um arquivo usando **SEARCH <nome_arquivo>**, o tracker verifica quais peers possuem esse arquivo e retorna a lista.

6. **Download**

   O download ocorre quando um peer solicita um arquivo disponível na rede. Nesse processo, o peer identifica quais outros peers possuem o arquivo e realiza a transferência em partes (*chunks*), possivelmente de múltiplas fontes ao mesmo tempo. Esse mecanismo permite paralelismo e maior velocidade.

   Os detalhes completos de implementação encontram-se no tópico “4 — Processo do download”.

7. **Desconexão de um Peer**
   Se o peer enviar **UNREGISTER** ou simplesmente parar de responder, o tracker:

   - detecta a ausência de heartbeats,
   - considera o peer inativo após o timeout,
   - e o remove da lista de peers ativos.

8. **Rede em Funcionamento Contínuo**
   Peers podem:

   - entrar e sair,

- compartilhar arquivos,
- buscar e baixar de múltiplos peers simultaneamente,
  enquanto o tracker sincroniza e mantém o estado global da rede.

3. Heartbeat :

O mecanismo de *heartbeat* foi implementado para permitir que o tracker identifique automaticamente quais peers continuam ativos na rede.
Quando um peer é iniciado, ele cria uma thread responsável por enviar periodicamente uma mensagem do tipo: HEARTBEAT <ip> <porta>. Esse envio ocorre em intervalos fixos e independentes das demais atividades do peer.

No lado do tracker, é mantido um dicionário chamado peer_last_seen, que registra a última vez que cada peer enviou um hea. Sempre que o tracker recebe a mensagem HEARTBEAT, ele apenas atualiza o tempo que o peer foi “visto” pela última vez.

Além disso, quando o tracker é iniciado, ele cria uma thread dedicada à verificação periódica desse dicionário. Essa thread compara o tempo atual com o último registro de cada peer. Caso o intervalo ultrapasse um *timeout* definido, o peer é considerado inativo.

Esse mecanismo garante que peers desconectados inesperadamente (por falha, desligamento ou encerramento abrupto do programa) não permaneçam listados como ativos no tracker.

4. Processo do download :

Quando um peer deseja realizar um download, o processo segue as seguintes etapas:

1. Busca pelo arquivo
O peer envia o comando search <nome_arquivo> pelo cliente.
O tracker retorna a lista de peers que possuem o arquivo solicitado.
Essa lista é armazenada localmente para ser utilizada no processo de download.

2. Solicitação de download
Em seguida, o peer executa o comando download <nome_arquivo> . A função download_file identifica o tamanho total do arquivo e calcula quantos **chunks** serão necessários.

CHUNK
Um *chunk* é um bloco de dados em que o arquivo é dividido.
A divisão em chunks permite baixar partes do arquivo de vários peers simultaneamente, aumentando a eficiência e reduzindo a dependência de um único peer, diminuindo o risco de baixar um arquivo malicioso.

3. **Distribuição dos chunks**
É criada uma fila contendo todos os chunks necessários para reconstruir o arquivo.
Para cada peer que possui o arquivo, é iniciada uma thread responsável por

consumir essa fila, solicitando chunks à medida que ficam disponíveis.

4. Reconstrução do arquivo

Quando todos os chunks são baixados e a fila fica vazia, o processo é encerrado.
Os chunks são então recombinados, formando o arquivo final.