Aluno: Jonas Ferreira da Trindade
Matrícula: 2016118533
E-mail: jonastrinda@gmail.com

Introdução

O objetivo do trabalho é criar um sistema de torrent. Nesse sistema os clientes desejam fazer download de um vídeo e por isso disparam requisições para os vários peers presentes na rede. Esses peers guardam partes desse vídeo (chunks) e o sistema deve controlar como o cliente chega até os peers, como os peer conversam entre si e devolvem respostas com essas partes do vídeo para os clientes.

Arquitetura

A arquitetura do sistema possui peers, clients, key-values-files, chunks e output logs. Os peers são responsáveis por guardar os chunks de acordo com o que está descrito nos key-values-files, conversar com os peers vizinhos e devolver uma resposta ao client sobre o chuns que possui. Os clients são responsáveis por iniciar a conexão com algum peer que está na rede, requisitar a ele os chunks que deseja fazer download e armazenar no output logs a resposta dos peers mesmo quando eles não possuírem os chunks solicitados.

Peer

- Armazena os chunks.
- Controla troca de mensagens entre peers.
- Recebe requisição do cliente ou dos outros peers sobre os chunks que o cliente deseja receber.
- peer <ip-local>:<local-port> <key-values-files_peer[id]> <ip1:port1> ... <ipN:portN> (seu ip, a porta que escuta, os peers que armazena, os ips vizinhos).
- Após o peer ser iniciado ele aguarda por mensagens.
- Um nó pode trocar mensagem com seus nós vizinhos.
- Cada chunk possui um id e um nome guardados na forma key:value, seu nome é uma string da seguinte forma BigBuckBunny_N.m4s onde N é o ID do chunk.
- Deve suportar várias consultas simultaneamente e transmitir chunks para diferentes clientes concorrentes.
- Resposta CHUNKS_INFO:3 com os peers que possuem os chunks solicitados
- Resposta RESPONSE
- Mensagem QUERY:2

Cliente

- Faz requisição ao peer falando os chunks que deseja receber.
- Utilizar socket.sendto para iniciar contato com peer
- Utilizar socket.recvfrom para aguardar respostas dos peers
- cliente <IP:port> <5,6,7> (endereço do porto de um peer, chunks que ele deseja receber do peer e dos seus peers vizinhos).
- Deve ser capaz de receber vários chunks de diferentes peers simultaneamente.
- Mensagem HELLO:1 para verificar quais peers possuem os chunks
- Mensagem GET:4

- Pode receber mensagens repetidas do mesmo chunk pois diferentes peers possuem chunks iguais.
- Função para estabelecer de qual peer baixar os chunks quando encontrar o chunk em diferentes peers.
- Não enviar mensagem GET:4 mais de uma vez para adquirir o mesmo chunk.
- Controlar os chunks consultados e recebidos.
- Guardar no output-IP.log as respostas com IP:porto do peer que enviou. IP no arquivo de log é o IP do cliente.
- Esse arquivo de log deve possuir M linhas contendo em cada uma peerIP:peerPort - chunkID.
- Quando nao encontrar um chunk deve guardar ‘0.0.0.0:0 - chunkID’
- Função timeout que define o tempo que ele aguarda por respostas. A inundação terminou.

Trocas Cliente - Peer

1. HELLO:1 (cliente: verificar quais peers possuem os chunks)
2. CHUNKS_INFO:3 (peer: responde quais peers possuem os chunks)
3. GET:4 (cliente: envia mensagem para os peers que possuem os chunks)
4. RESPONSE

Trocas Peer - Peer

1. QUERY:2 (peer: envia uma query para os peers vizinhos)

Discussão

O trabalho foi muito interessante para entender melhor como um sistema torrent deve funcionar. Antes entendia que compartilhava arquivos a todo momento quando usava esse tipo de aplicação mas não entendia profundamente como funcionava esse compartilhamento de arquivos. Não foi possível implementar as principais funções porém criei pseudocódigos com a lógica que implementaria, esse semestre foi muito difícil, estou trabalhando presencial mesmo na situação que estamos vivendo e às vezes é muito difícil encontrar tempo para estudar e fazer os trabalhos práticos. Espero que entendam minha situação. Obrigado.