Relatório TP3
Redes de Computadores

João Marcos Machado Couto
Matrícula:201714421

Março 2021

1 Introdução

Sob o desafio de implementar um sistema peer-to-peer para compartilhamento de vídeos, este trabalho prático provou-se desafiador na medida que ao contrário dos outros TPs até então, as comunicações não são apenas de ida e volta, não, aqui foi necessária a implementação de lógicas para viabilizar a comunicação não só entre cliente e peers mas também do peers entre si via uma lógica de flooding de queries

2 Overview da implementação: fluxo típico de comunicações

Uma tranmissão do vídeo se inicia no momento em que o cliente manda uma mensagem HELLO via UDP para seu peer "ponto-de-contato" (POC) na rede peer-to-peer. Nessa mensagem, ele informa quais são os chunks do vídeo que ele deseja baixar. Apartir deste momento clientes ficam até 5 segundos aguardando respostas do tipo CHUNK INFO dos peers. Simultaneamente, o peer POC extrai o endereço da porta UDP do cliente que mandou a requisição via o retorno da função recvfrom e, então, prontamente manda uma mensagem do tipo QUERY para seus peers vizinhos, os informando do endereço do cliente e também a lista de chunks que ele está interessado em receber. Os vizinhos por sua vezes retransmitem a query para seus outros vizinhos contanto que o TTL recebido por ele seja maior que 1.

Durante o periodo de 5 segundos antes do timeout o cliente então processa todas as mensagens CHUNK_INFO que receber dos peers e após o timeout escolhe,aleatoriamente, para cada chunk, um peer dentre aqueles cujo chunk info constava a disponibilidade do chunk.

Escolhido o peer para cada chunk de interesse que de fato estavam presentes em algum dos peers que conseguiriam se comunicar com o cliente, ele efetiva então transmissões do tipo GET para cada um dos peers escolhidos, informando o interesse em receber daquele peer o chunk para qual ele foi selecionado. Recebido o GET, cada peer então responde ao cliente com uma transmissão do tipo RESPONSE

3 Overview da implementação: decisões aplicaveis a peers e clientes

Tanto no cliente quanto nos peers optei por determinar a fonte de uma transmissão utilizando o segundo retorno da função recvfrom que informa, para cada mensagem retirada do buffer do UDP, qual é o endereço IP e porta da fonte da mensagem.

Como a grande massa dos campos prensentes nos segmentos dos diferentes tipos de transmissões do sistema são inteiros, o "work horse" principal nas construções das mensagems transmitidas tanto dos peers quanto dos clientes foi a função to_bytes de inteiros em python, que recebe o número de bytes que o inteiro deve ocupar e faz o padding de zeros necessário para assegurar que a transformação do inteiro em bytes ocupe o número de bytes indicado no parametro. Naturalmente, a decodificação desses inteiros é feita utilizando a função int.from_bytes() que faz o processo contrário.

4 Estruturas de dado e lógicas da implementação:

- Cliente:
  - *desiredChunks:* trata-se de uma lista de inteiros que em cada cliente armazeada os índices dos chunks que o cliente deseja receber. É utilizado em dois momentos principais no código. Primeiramente é utilizado na construção da mensagem HELLO enviada para o peer POC do cliente. Posterior é utilizado também para definir as chaves do dicionário chunkPeers descrito abaixo

  - *chunkPeers:* trata-se um dicionário indexado pelos índices dos chunks que o cliente deseja receber presentes na lista desiredChunks. Para cada um desses chunks, este dicionário armazena uma lista de tuplas (IP, porta) onde cada tupla representa um peer que informou ao cliente via um transmissão CHUNK_INFO que tem este chunk disponível para requisições via GET. É utilizado em dois momentos: o primeiro é quando o cliente está recebendo pacotes CHUNCK_INFO que são processados para identificar em quais posições do dicionario cada peer deve entrar de acordo com a lista de chunks presente em seu CHUNK_INFO . Posteriormente o chunkPeers é utilizados para decidir aleatoriamente para qual peer as requisições GET serão enviadas para cada chunk

  - *peerIndexPerChunk:* uma lista que guarda o índice do peer dentro de cada lista de peers associada com um chunk no dicionário chunkPeers, que sera utilizado como destino da requisição GET que posteriormente sera chamada para cada chunk desejeado pelo cliente. A construção dessa lista é randomica: para cada chunk selecionamos um peer qualquer dentre as opções presentes no chunkPeers e guardamos o indice dele nesta lista.

- Peer:
  - *localStorage:* trata-se de um dicionário indexado pelos índices de chunks que um dado peer tem disponível localmente. Cada indice de chunk é então mapeado para uma string que guarda o nome do arquivo associado com o chunk. É utilizado para verificar quais arquivos estão presentes no armazenamento local do peer

  - *chuncksInStorage:* uma lista utilizada na construção de mensagens de CHUNK_INFO: dada uma lista de chunks desejados por um cliente, seja via QUERY, seja via HELLO, o peer verifica a presença de cada chunk nas chaves do dicionário localStorage. Os chunks que estiverem presentes estão disponiveis localmente no peer portanto têm seus índice apendados a esta lista. Assim ao final temos uma lista de chunks para os quais houve casamento entre os chunks requisitados por um cliente e os chunks presentes no peer.

  - *chunckList:* apenas uma lista auxiliar utilizada para armazenar os índices dos chunks extraídos de uma query. É utilizada para iteramos sobre os chunks da query e então verificar se estão presents no localStorage do peer, auxiliando na construção de transmissões do tipo CHUNCK_INFO