FUNDAÇÃO UNIVERSIDADE FEDERAL DO ABC

KAREN OLIVEIRA DOS SANTOS RA: 11110915

PROJETO Napster DA DISCIPLINA DE SISTEMAS DISTRIBUíDOS

SANTO ANDRÉ - SP

2022

1. Link do vídeo

2. Funcionamento do Servidor

Servidor recebe o IP por entrada do próprio usuário e abre uma porta UDP para tratar requisições (10098)[Linha 38 Servidor] e uma Thread para tratar o Alive (10098)[Linha 39 Servidor].

Devido a estrutura do código fonte, citarei as Threads de acordo com a variável funcao.

Os peers são identificados pelo seguinte formato: [IP];[PORTA_UDP];[PORTA_TCP_1:PORTA_TCP_2], que será referido como chave do Peer.

2.1 ALIVE/SEND-ALIVE

O ALIVE [Linha 82 - 94] é responsável por gerenciar o envio do ALIVE para os Peers, incluindo a pausa de 30 segundos na linha 85. Para cada peer o ALIVE executa o SEND-ALIVE [Linhas 96 - 116], esse é responsável por enviar a requisição para cada Peer, caso tenha um Timeout ou o Peer não responda ALIVE_OK, o Servidor retira o Peer da sua lista de Arquivos.

2.2 ESCUTAR-UDP/PROCESSAR-UDP

O ESCUTAR-UDP [Linha 70 - 80] é a escuta do Socket UDP onde o Servidor irá receber suas Requisições UDP, porém não será ele a função responsável por processar a mensagem, ele delega para o PROCESSAR-UDP [Linha 118 -158].

O PROCESSAR-UDP recebe a mensagem e realiza o procedimento de acordo com a variável Action da Mensagem.

2.2.1 JOIN

O JOIN [Linha 120 - 125] recebe a requisição do Peer com as Portas (TCP e UDP) e os Arquivos daquele Peer. Ele se encarrega de gerar a chave do Peer e salvar os arquivos na Map (fileByServer). A chave é a chave do Peer e o valor é a lista de arquivos que aquele Peer hospeda.

2.2.2 LEAVE

O LEAVE [Linha 126 - 130] é o inverso do JOIN, após montar a chave, ele remove o arquivo do fileByServer. Ele recebe os mesmos dados do JOIN.

2.2.3 SEARCH

O SEARCH [Linha 131- 139] recebe o nome do Arquivo, o Servidor itera pelas chaves de Peer do fileByServer, montando uma lista de chaves de Peers que tenham o nome de Arquivo enviado pelo Peer e retorna a lista.

2.2.4 UPDATE

O UPDATE [Linha 140 - 148] recebe o nome do arquivo e o Peer que recebeu ele, ele monta a URL do Peer e realiza a adição do nome do Arquivo novo na lista do que o Peer já tinha.

3. Funcionamento do Peer

3.1 JOIN

O JOIN [Linha 30 - 79] recebe dados do usuário no padrão:

JOIN [PATH_PASTA_PEER] [PORTA_UDP] [PORTA_TCP_1] [PORTA_TCP_2]

PATH_PASTA_PEER sendo o path da pasta onde fica os arquivos desse Peer.
PORTA_TCP_X deve ter no mínimo uma Porta TCP.

Exemplos:
JOIN C:\Users\karen\Documents\peer1 localhost 9001 9091 9094
JOIN C:\Users\karen\Documents\peer2 localhost 9002 9092

Itera pela pasta PATH_PASTA_PEER salvando os nomes dos Arquivos contidos nelas e envia os dados para o Servidor.

3.2 LEAVE

O LEAVE[Linha 80 - 87] recebe dados do usuário no padrão:

LEAVE

Envia os dados do Peer ao Servidor e aguarda o LEAVE_OK.

3.3 UPDATE

Não é uma requisição enviada pelo usuário, ela só é feita quando o Peer faz o Download de algum arquivo de forma bem sucedida, o próprio Download faz a chamada ao UPDATE [Linha 291 - 294]. Ele envia os dados do Peer e o nome do Arquivo novo para o Servidor

3.4 SEARCH

O SEARCH[Linha 88 - 99] recebe dados do usuário no padrão:
SEARCH [NOME_ARQUIVO]

NOME_ARQUIVO é o nome do Arquivo a ser procurado.
Ele envia o nome do Arquivo ao Servidor.

3.5 DOWNLOAD

3.5.1 REQUEST-DOWNLOAD [Peer Origem da Requisição]

O DOWNLOAD [Linha 88 - 99] recebe dados do usuário no padrão:
DOWNLOAD [IP_DESTINO] [PORTA_TCP_DESTINO] [NOME_ARQUIVO]

NOME_ARQUIVO é o nome do Arquivo requisitado.
IP_DESTINO IP do Peer para qual o Arquivo será requisitado.
PORTA_TCP_DESTINO do Peer para qual o Arquivo será requisitado.

É chamado o REQUEST_DOWNLOAD [Linha 246 - 290] é a função responsável por requisitar, receber e confirmar que o arquivo está correto por um Hash MD5.

O Peer origem envia um Request com função Download contendo o nome do arquivo que deseja. O primeiro dado pode ser um DOWNLOAD_NEGADO ou o Hash do Arquivo, para diferenciar é realizado uma tentativa de ler o json [Linha 260], caso não seja o código entende que é o Hash do Arquivo, após o Hash é recebido o tamanho do Arquivo e o arquivo em si. Após receber o arquivo é comparado o Hash dele com o enviado pelo Peer que enviou o arquivo [Linha 276]. Caso os Hashes sejam iguais, o entende-se que o arquivo está correto e é realizado o UPDATE.

Caso o Download não ocorra, não foi implementado nenhuma tratativa.

3.5.2 ESCUTAR-TCP [Peer Destino da Requisição]

Recebe o nome do arquivo , verifica se tem o arquivo solicitado [Linha 221], verifica se vai negar aleatoriamente [Linha 226 - 227]. Caso negue, realiza o envio da String DOWNLOAD NEGADO. Caso aceite, será enviado o Hash, tamanho e o próprio arquivo solicitado.

3.6 ALIVE

O ALIVE é começado na Linha 55, porém a função responsável por processar o ALIVE vindo do Server é nas Linhas 195 -207.Somente retorna ALIVE_OK para as requisições recebidas.

4. Funcionamento da Thread

O construtor da Thread tem o Atributo funcao, ele define o que a Thread será responsável por fazer.

O Servidor é composto por duas Threads principais que usam Threads secundárias para processar a informação.

- ALIVE realiza o gerenciamento do ALIVE, criando uma Thread secundária SEND-ALIVE para enviar os ALIVES ao Peers e entrando em sleep até o próximo batch de ALIVE.
- ESCUTAR-UDP, ela fica responsável por escutar o Socket UDP do Servidor chegando alguma requisição, ele cria uma Thread PROCESSAR-UDP e despacha para esta processar de Fato a requisição.

Já para o Peer é criado uma Thread simples para o ALIVE assim que é feito o JOIN, Threads para escutar os Sockets TCP que foram passados no JOIN, essas Threads ficam em Requisição todo momento, porém quando o Peer requisita um Download é criada uma nova Thread (REQUEST-DOWNLOAD) para tratar de baixar o arquivo.

5. Implementação da transferências de arquivos grandes

Na parte de envio do Arquivo Linhas 328 - 348 é feito o envio do tamanho do Arquivo e após o próprio arquivo, esse arquivo é enviado em pedaços pequenos [Linha 343].

Na parte de recebimento [Linhas 305 - 318], o buffer permanece aberto pelo tamanho do Arquivo que foi passado pelo remetente do arquivo [Linha 313].

Links

https://heptadecane.medium.com/file-transfer-via-java-sockets-e8d4f30703a5
Links passados na própria atividade.