Universidade Federal de Pernambuco

Centro de Informática

Curso de Graduação em Ciência da Computação

Infra-Estrutura de Comunicação

Professor: Paulo André da S. Gonçalves

**Projeto de Infra-Estrutura de Comunicação: Trololotorrent**

**Bruno Machado Dias Macena (bmdm)**

**Igor Gomes (ilrg)**

**Jonathas Mendes Alves (jma)**

**Recife, julho de 2010**

**Bruno Machado Dias Macena (bmdm)**

**Igor Gomes (ilrl)**

**Jonathas Alves (jma)**

**Projeto de Infra-Estrutura de Comunicação: Trololotorrent**

Trabalho complementar ao seminário que serve como pré-requisito para obtenção de nota na disciplina de Infra-Estrutura da Comunicação no curso de Ciência da Computação, sob a orientação do professor Paulo André da S. Gonçalves.

**Recife, julho de 2010**

Sumário

[1. Linguagem e softwares adotados 2](#_Toc265766587)

[2. Protocolos 2](#_Toc265766588)

[2.1. Protocolo entre peer e tracker (HTTP) 2](#_Toc265766589)

[2.2. Protocolo entre peer e servidor (HTTP) 3](#_Toc265766590)

[2.3. Protocolo entre peers (Trolocolo) 3](#_Toc265766591)

[3. Arquivo ***.torrent*** 6](#_Toc265766592)

[4. Tracker 6](#_Toc265766593)

# Linguagem e softwares adotados

A linguagem utilizada foi JAVA e utilizamos os programas ECLIPSE e NETBEANS.

# Protocolos

## Protocolo entre peer e tracker (HTTP)

O protocolo utilizado para comunicação entre peer e tracker será o HTTP/1.1. Os tipos de mensagem utilizados serão: GET, POST, HEAD e DELETE.

Quando o cliente envia uma mensagem GET (Ex: GET Avatar HTTP/1.1), o tracker enviará uma resposta HTTP contendo a lista de todos os peers e seeds cadastrados no mesmo para distribuir aquele arquivo.

A mensagem POST serve para o cliente indicar ao tracker que quer distribuir um novo arquivo. Ela contem apenas o nome POST e o MD5 do arquivo. Dessa forma o tracker irá guardar a informação desse novo arquivo que será distribuído na rede.

O HEAD serve para o peer enviar uma mensagem informando que continua disponibilizando o arquivo. Essa mensagem é enviada a cada 5 minutos pelo peer. Se o tracker passar mais de 5 minutos sem recebê-la de um peer que está disponibilizando arquivos, seu IP será retirado da lista de peers ativos.

Utilizamos DELETE para o peer informar ao tracker que não está mais disponibilizando aquele arquivo.

## Protocolo entre peer e servidor (HTTP)

O servidor disponibiliza uma página web onde será possível fazer o upload do arquivo *.torrent* e realizar buscas dos arquivos que estão no servidor. Além disso, é possível fazer o upload do arquivo *.torrent* diretamente pelo programa Trololorrent.

Para fazer o envio do .torrent acessamos a página web *http://<endereço IP do servidor>:<nº da porta>/upload.html*. Esse upload também pode ser feito através do Trololorrent de forma transparente para o usuário.

 Para buscar um arquivo no servidor acessamos a página *http://<endereço IP do servidor>:<nº da porta>/busca.html?q=<nome do arquivo>*. O servidor irá retornar uma lista de arquivos (se possuí-lo) e será possível baixar o seu arquivo .torrent referente.

## Protocolo entre peers (Trolocolo)

O protocolo trolocolo foi projetado para gerar o menor overhead possível.

Toda a comunicação será feita em bytes. Se por exemplo quisermos informar o tamanho do dado enviando, informaremos o inteiro **1678**. Assim fica bem menos custoso do que usar Strings. Não haverá quebras de linhas, já que o receptor saberá exatamente a quantidade de bytes enviados.

1 16 4 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TIPO | LENGTH | PIECE | OFFSET | DADOS |

\* tamanho do campo em bytes

Mesmo que a mensagem seja do tipo HANDSHAKE e não use length, piece ou offset, vamos ter este espaço reservado para eles.

Os tipos agora são inteiros e cada um está associado a um número.

0 - HANDSHAKE

1 - BITFIELD

2 - INTERESTED

3 - NOT INTERESTED

4 - CHOKE

5 - UNCHOKE

6 - REQUEST

7 - PIECE

8 - HAVE

9 – CANCEL

 Fluxo de conexão entre peers

Handshake

Handshake

Bitfield

Bitfield

Interested

Unchoke

Request

Piece

Request

Piece

...

...

Request

Rave

Adição do tipo **CLOSE**.

Quando uma mensagem CLOSE é enviada, avisamos ao outro peer o interesse de fechar a conexão. A conexão só será fechada após o outro peer responder com um CLOSE.

Um peer que mandar uma mensagem CLOSE não pode fazer qualquer requisição (embora possa responder a requisições do outro peer). Se quiser fazer uma requisição, terá que mandar um novo HANDSHAKE.

O peer que recebeu um CLOSE pode continuar fazendo novas requisições para o peer que mandou esta mensagem. A conexão só é fechada caso os 2 peers enviem um CLOSE.

Temos que ter este tipo de mensagem se quisermos fechar uma conexão de maneira segura. Temos ainda que pensar num tipo de mensagem que feche a conexão mesmo que o outro peer queira baixar novos arquivos de você.

E esse CLOSE que eu sugeri é pro peer avisar ao outro que ele tá **com vontade** de fechar a conexão. Se a outra ponta quiser, pode fechar a vontade.

Pq se as 2 pontas não quiserem mais usar aquela conexão, como eles vão avisar um ao outro que querem fechá-la?

Fluxo para um usuário iniciar o download de um arquivo

Ou

Transparente p/ o usuário

Fluxo para enviar “.torrent” ao tracker

# Arquivo ***.torrent***

O nome do arquivo .torrent vai ser a soma MD5 do conteúdo do arquivo. Ex: um arquivo de texto cujo conteúdo seja "123" teria seu MD5 igual a ***202cb962ac59075b964b07152d234b70***

Dentro do arquivo:

* O IP do tracker e sua porta. Ex: *192.168.0.100:9001*
* O nome do arquivo original
* O tamanho em bytes do arquivo

# Tracker

Quando o cliente abre o arquivo .torrent, ele requisita uma página (via HTTP) ao tracker. O nome da página é o MD5 do arquivo que se deseja baixar Ex: *http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70*

O tracker responde com a lista de peers conectados com o arquivo.

Quando o cliente é aberto, o passo anterior é feito novamente. Ele faz essa mesma requisição para cada arquivo que tem.

A cada 5 minutos, o *peer* informa ao *tracker* que está vivo (apenas uma vez, mesmo que ele tenha vários arquivos disponíveis). Essa informação é enviada via HTTP.

Ex: *http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70/keepalive*

O cliente pode mandar a mensagem **remove** para o tracker.

*Ex:*[*http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70/remove*](http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70/remove)

Quando o cliente completa o download, se ele quiser continuar compartilhando, ele avisa ao tracker.

*Ex:*[*http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70/completed*](http://192.168.0.100:9001/202cb962ac59075b964b07152d234b70/completed)