Gateway Aplicacional e Balanceador de Carga sofisticado para HTTP

Adriana Gonçalves a75119, Bernardo Viseu a74618, and Marco Gonçalves a75480

Universidade do Minho, Portugal

Abstract. No âmbito da unidade curricular de Comunicação por Computadores foi-nos proposto a elaboração de um trabalho prático desenvolvido em JavaScript, cuja finalidade é implementar um gateway de aplicação (HttpGW) que opere exclusivamente com o protocolo HTTP/1.1 e que seja capaz de responder a múltiplos pedidos em simultâneo, recorrendo a FastFileSrv.

Keywords: HTTP · TCP · UDP · JavaScript · Servidor · FastFileSrv

1 Introdução

Este relatório é relativo à elaboração do segundo trabalho prático proposto pela unidade curricular de Comunicação por Computadores.

Para a realização deste trabalho, foi necessário reflectir sobre como iríamos implementar todas as funcionalidades pretendidas da melhor forma, bem como qual a linguagem de programação utilizar para escrever o programa.

Ao longo do relatório iremos explicar todas as decisões e abordagens tomadas ao longo do desenvolvimento do projecto, de forma a obter o resultado final desejado.

2 Arquitectura da Solução

Para a realização deste projecto, primeiro foi necessário pensar num protocolo para ser utilizado na comunicação entre o *HttpGW* e os *FastFileSrv*. Acabamos por concluir que iríamos fazer um método baseado em cada FastFileSrv enviar uma mensagem ao gateway com um formato específico, comunicando ao gateway o seu endereço e porta, assim como os ficheiros que este servidor tem disponíveis para descarregamento.

Depois disto já decidido, pensamos em que linguagem de código iríamos usar para a implementação. Após consideração de algumas linguagens comuns, decidimos utilizar *JavaScript*, devido à simplicidade de código e existência de *libraries* úteis para este trabalho.

3 Especificação do protocolo

3.1 Formato das mensagens protocolares

Para este projecto foi necessário definir um método de um *FastFileSrv* comunicar com o *HttpGW* de maneira a indicar a sua porta e endereço, assim como a lista de ficheiros que este servidor consegue enviar para o *Gateway*.

Para tal, o método que escolhemos foi o envio de uma mensagem no momento de início de um servidor *FastFileSrv* com o seguinte formato:

```
PORT:[PORTA FFS]::::FILELIST:[FILE1]++[FILE2]++[FILE3]++[...]
```

O *HttpGW* reconhece esta mensagem e faz *parse* da mesma de maneira a guardar num *array* um *key, value* em que a *key* é a porta do *FFS*, e o *value* é a lista de ficheiros que o *FFS* tem disponíveis.

Utilizando este *array*, o *HttpGW* consegue decidir qual *FastFileSrv* utilizar para pedir um certo ficheiro, e também é assim que identifica se um ficheiro existe em algum servidor, e caso não exista envia uma mensagem de erro.

3.2 Interacções

Neste projecto existem várias interacções, entre o *HttpGW* e o(s) *FastFileSrv*, assim como os pedidos de transferência de ficheiros efectuados por um utilizador para o *Gateway*.

Iniciado o *HttpGW*, este vai disponibilizar uma rota *HTTP* assim como uma ligação *UDP*. Um *FastFileSrv* que seja inicializado vai comunicar ao *Gateway* os seus dados (como foi indicado na secção anterior). Um utilizador pode então fazer pedidos de transferências de ficheiros utilizando, por exemplo, o comando *wget*. Isto faz um pedido GET para o servidor HTTP que temos a correr em *HttpGW*, que vai então verificar se o ficheiro pedido está disponível para descarregar de algum *FastFileSrv* inicializado. Se algum *FFS* tiver o ficheiro disponível, o *Gateway* envia lhe uma mensagem *UDP* com o nome do ficheiro requerido, e aguarda por uma resposta, que vai ser composta por um conjunto de chunks enviados separadamente que representam o ficheiro pedido. Por fim, o *Gateway* envia o ficheiro inteiro para o utilizador que o pediu.

4 Implementação

Para a implementação deste projecto começamos por desenvolver um servidor HTTP. Para isto, usamos a ferramenta *Express*, uma ferramenta *web framework* capaz de gerar código para um servidor HTTP simples.

Este servidor foi feito para receber os pedidos de descarregamentos (por exemplo, utilizando *wget*).

Utilizando o código gerado pelo *Express* como base, conseguimos desenvolver um servidor UDP capaz de enviar e receber mensagens aos servidores *FastFileSrv*. Temos então um *HttpGW* que está a espera de pedidos numa porta especificada, e assim que recebe um pedido de um ficheiro vai verificar primeiro

se algum *FastFileSrv* tem esse ficheiro disponível e, caso algum o tenha, vai descarregar esse ficheiro por *chunks* de 1024 *bytes*. Estes chunks são enviados por UDP para o *HttpGW*, e finalmente é enviado para o utilizador que fez o pedido original.

O ficheiro *fastfilesrv.js* tem todo o código dedicado às funcionalidades do *FastFileSrv*, ou seja, uma ligação UDP que comunica com o *HttpGW* (para isto é necessário indicar como argumentos o endereço e porta em que o servidor UDP *HttpGW* está a correr). Ao iniciar um servidor *FFS*, este vai enviar uma mensagem inicial com um formato específico já indicado anteriormente, e depois disto já está pronto para receber pedidos do *HttpGW*. É também neste ficheiro que se faz a separação de ficheiros por *chunks* antes de serem enviados para o *HttpGW*.

4.1 Detalhes, parâmetros, bibliotecas de funções, etc..

Para este trabalho utilizamos código *Javascript*, com o auxílio da *library Express* (para o servidor *HTTP*), assim como a *library DGRAM* necessária para comunicações *UDP* nesta linguagem de programação (utilizada no *HttpGW* assim como *FastFileSrv*).

5 Testes e resultados

Para correr o *HttpGW* corremos "npm install" "npm start" ou "nodemon" na pasta HttpGw.

Tendo o Gateway iniciado precisamos de iniciar os *FastFileServer* e estes servidores vão partilhar todos os ficheiros que se encontrem dentro da pasta onde ele é executado. Para o executar podemos utilizar o seguinte comando "node fastfilesrv.js 10.1.1.1 53016" temos que executar o código em JavaScript que recebe 2 parâmetros como argumento o 1º é o ip do gateway e o segundo é a porta que este comunica com os *FastFileSrv*.

Fig. 1. HttpGW iniciado no Server1

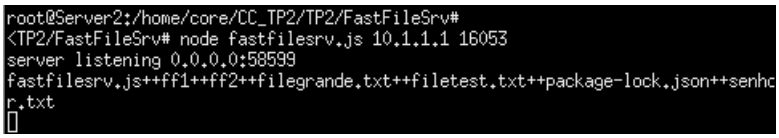

Fig. 2. FastFileServ iniciado no Server2

Fig. 3. *HttpGw* após o inicio de 2 *FastFileSrv*

Para testar executamos alguns wget a partir do Laptop2 "wget http://10.1.1.1:3000/ff1.txt" o nosso *HttpGW* está á escuta dos pedidos na porta 3000 e como podemos verificar com o comando ls os nossos programas funcionaram.

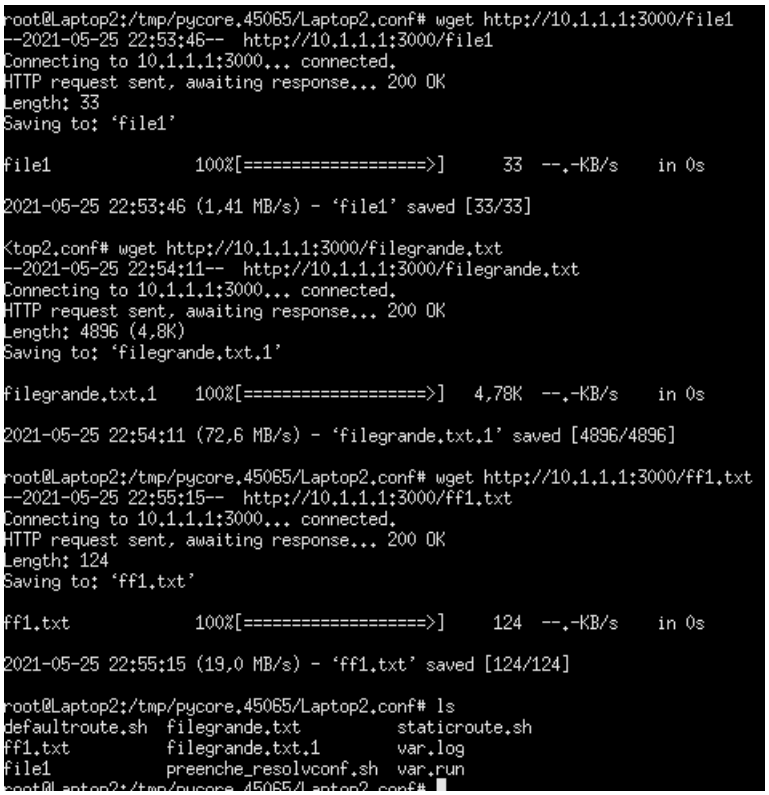

Fig. 4. Laptop2 wget's de diferentes ficheiros

6 Conclusão e trabalho futuro

Este trabalho ajudou o grupo a compreender melhor as formas de comunicação entre um utilizador e um servidor. Não conseguimos implementar o Keep Alive dos FastFileSrv. Foi um estudo muito útil na aprendizagem de ligações *UDP* na linguagem *Javascript*, algo que até agora nos era desconhecido.

Ficamos satisfeitos com o resultado final deste projecto, sabendo que temos vários elementos que poderiam ser melhorados, como um melhor método de identificação de servidores *FastFileSrv* que foram interrompidos, assim como incluir de encriptação nos pacotes enviados de maneira a tornar as redes mais seguras, até mesmo a introdução de um keep alive.