Relatório do projeto de Introdução às Redes de Comunicação 2019/2020

Duarte Dias 2018293526
Gabriel Fernandes 2018288117

O programa é constituído por três componentes, o cliente, o servidor e o proxy. A utilização do proxy é opcional, se este não se encontrar em funcionamento, as comunicações serão efetuadas diretamente entre cliente e servidor.

Na nossa solução utilizámos TCP para as comunicações realizadas aquando do pedido de LIST e QUIT por parte do cliente. Se o pedido for de DOWNLOAD a aplicação utiliza TCP ou UDP.

Funcionamento do Cliente

O cliente recebe pela linha de comandos o ip do servidor e do proxy e o porto ao qual se deve conectar. Se os dados forem bem introduzidos o cliente prossegue e cria um socket tcp, conecta-se ao servidor (se o proxy estiver em funcionamento a ligação é efetuada com o proxy) e fica pronto para receber comandos a partir do terminal.

Consoante o comando que lhe é passado, este pode listar todos os ficheiros presentes no servidor para download ('LIST'), fazer o download de um ficheiro do servidor ('DOWNLOAD <TCP/UDP> <NOR/ENC> <nome do ficheiro>') ou terminar a ligação com o servidor ('QUIT').

Funcionamento do Proxy

O proxy, quando em funcionamento, gere as ligações entre os clientes e o servidor, cria um socket UDP e um socket TCP onde vai receber as novas comunicações. Quando lhe chega uma nova ligação este cria uma thread que fica encarregue das comunicações com este cliente. Esta thread recebe, do cliente, as informações necessárias para se conectar ao servidor, cria o socket e conecta-se a este.

Após as conexões entre cliente <-> proxy e proxy <-> server estarem estabelecidas o cliente pode começar a enviar comandos para o servidor (intermediados pelo proxy) e a receber informação enviada por este.

Funcionamento do Servidor

O servidor, quando é inicializado cria um socket UDP e um socket TCP onde vai receber as ligações vindas do proxy (se este estiver ligado), ou do cliente. Quando lhe chega uma nova ligação, este cria um processo para lidar com o cliente que chegou.

O novo processo fica à espera de receber comandos do cliente que chegou (quer este seja o proxy ou o cliente diretamente, para o servidor são ambos clientes). Quando recebe um comando executa a ação adequada, se for um download o server envia o ficheiro pelo protocolo solicitado pelo cliente (se a ligação for recebida por um socket UDP o ficheiro é enviado usando o protocolo UDP, se for recebida por um socket TCP o protocolo usado é TCP).

Funcionalidades implementadas:

TCP (tanto com proxy como sem proxy):
- Listar os ficheiros do servidor;
- Fazer o download desencriptado dos ficheiros do servidor;

UDP(apenas funciona para um cliente ao mesmo tempo):
- Listar os ficheiros do servidor;
- Fazer o download desencriptado dos ficheiros do servidor;

Proxy:
- Permite mostrar as informações das ligações que estão a decorrer entre o cliente e o servidor;

Cliente:
- Permite terminar a ligação com o servidor;
- Enviar os comandos pretendidos, à excessão do download com UDP com o proxy em funcionamento.

Funcionalidades não implementadas:

TCP:
- Não permite fazer o download de ficheiros encriptados;

Proxy:
- Não permite efetuar losses aos bytes enviados via UDP
- Não permite o salvamento do último ficheiro enviado pelo servidor ao cliente;

Dificuldades:

Tivemos dificuldade a implementar a transferência de informação por via do protocolo UDP, desde como implementar o cliente para este saber quanto deveria ler do socket (devido à possibilidade de ocorrer perda de bytes) até a estruturação do proxy para funcionar com UDP(quantos sockets deveríamos ter, por exemplo).

O que fizemos:

Acabámos por fazer a ligação UDP a funcionar para apenas um cliente, o que está longe do ideal, visto que a aplicação é um servidor.

O que queríamos ter feito:

No UDP pensámos em enviar, no início da mensagem, o número de bytes que o cliente deveria ler (efetuaríamos a perda de bytes no final da mensagem) . Pensámos também em usar apenas um socket UDP no proxy para gerir as comunicações vindas tanto dos clientes como do servidor, tal e qual como a informação enviada pelo proxy para ambos. Para isso, colocaríamos no início da mensagem o ip do servidor na mensagem enviada pelo cliente ao proxy e o proxy enviaria a mesma mensagem, mas colocaria o ip do cliente na mensagem enviada para o servidor. Assim, o proxy saberia que a comunicação tinha como destino o servidor e enviá-la-ia para este, caso contrário, se o ip não fosse o do servidor, o proxy enviaria a mensagem para o cliente correspondente ao ip contido na mensagem. Para a ligação ser possível precisaríamos do porto do cliente, informação que era retirada das lista guardada no proxy que contém informação da origem e destino de todas as ligações em curso(retiraríamos o porto do cliente, ou seja, da origem).

Não conseguimos implementar esta parte final, porque estávamos a ter problemas a receber as mensagens vindas do servidor para o proxy.

Detalhes de implementação

-> O proxy recebe o ip do servidor através de uma mensagem enviada para este pelo cliente, mas nós temos o ip do servidor no proxy também (guardado numa variável global) pois, devido à nossa implementação, precisávamos dessa informação antes da mensagem que contém o ip do servidor chegar ao proxy;
-> O porto do cliente, servidor e proxy é o mesmo;
-> Para sabermos que a listagem dos ficheiros do servidor chegou ao fim utilizámos uma mensagem que diz ao cliente que a listagem acabou;
-> Nos downloads, enviamos uma mensagem quer ao proxy, quer ao cliente com o número de bytes que o ficheiro ocupa, assim, estes contam o número de bytes que recebem e comparam com o tamanho do ficheiro, se este for o mesmo significa que receberam o ficheiro completo e podem terminar o processo de leitura;