Trabalho de Aprofundamento 2: Speedtesting

UNIVERSIDADE DE AVEIRO

João Tiago Rainho, Sofia Teixeira Vaz

VERSAO

Trabalho de Aprofundamento 2: Speedtesting

DEPARTAMENTO DE ELETRÓNICA,
TELECOMUNICAÇÕES E INFORMÁTICA

UNIVERSIDADE DE AVEIRO

João Tiago Rainho, Sofia Teixeira Vaz
(92984) tiago.rainho@ua.pt, (92968) sofiateixeiravaz@ua.pt

31/03/2019

Resumo

Este trabalho, que se baseia na criação de um cliente para fazer *speedtesting*, visa a proteção do utilizador, como foi referido na introdução. No entanto, a nossa ferramenta, apesar de eficiente, não é o melhor disponível, uma vez que diversas empresas já fizeram ferramentas parecidas com resultados mais viáveis.

Conteúdo

Lista de Tabelas

Capítulo 1

Introdução

A internet, apesar de ser vendida como algo tangível, não o é. Assim, o consumidor pode facilmente ser enganado pela operadora, uma vez que pode estar a pagar por uma qualidade que não está a receber e nem se aperceber disso. É nesse contexto que a nossa ferramenta de *speedtesting* foi criada, no entanto, há mais razões. Uma destas pode ser verificação de tráfego, e outra ainda é verificação da qualidade da placa de rede do computador em si. Este documento está dividido em quatro capítulos. Depois desta introdução, no Capítulo 2 é apresentada a metodologia seguida, no Capítulo 3 são apresentados os resultados obtidos, sendo estes discutidos no Capítulo 4. Finalmente, no Capítulo 5 são apresentadas as conclusões do trabalho.

Capítulo 2

Metodologia

Na fase inicial, tentámos perceber como dividir melhor o trabalho para se conseguir fazer o código da maneira mais simples possível. Para isto, usámos os pontos especificados no guião, trocando a ordem de alguns. Ainda adicionámos algumas features que não são mencionadas no guião. Por exemplo, quando o nome do país inserido não existe no ficheiro de servidores, ligamo-nos a um servidor qualquer da lista.

2.1 Cálculo da latência

Para calcular a latência, calculámos o tempo que o server demora a responder PONG após receber PING. Neste cálculo, decidimos ignorar o primeiro PING-PONG, uma vez que esse valor seria sempre muito maior do que os outros. Assim, em vez de correrm o ciclo 11 vezes (para calcular 10 latências), corremos o ciclo 12 vezes.

2.2 Cálculo da largura de banda

Decidimos que a melhor maneira de calcular a largura de banda seria receber 1 MB de informação (para abrir os portos de informação), enviar a restante informação (INFO - 1 MB) e ver quanto tempo isso demorou (t). Assim, o cálculo seria

$$\frac{\text{INFO - 1 MB}}{\text{t}}$$

2.3 Modulação do código

Após fazermos o código necessário para o cálculo da largura de banda, apercebemo-nos de um problema - o nosso código não era modulável de maneira nenhuma. Assim, decidimos pegar no código que tínhamos e fazer com que cada parte fosse uma função. Assim, cada função poderia ser usada sozinha num outro

programa qualquer. É de frizar que fazer um programa altamente modulável é uma boa prática de programação e, além disso, é mais fácil testar no que toca à procura de *bugs* e resolução desses mesmos.

2.4 Escrita dos documentos

Para a escrita, decidimos fazer tudo da maneira mais simples, invocando as funções de cálculo de latência e de cálculo de largura de banda dentro da função que escreve o ficheiro em si. Decidimos abrir o ficheiro em modo "a"em vez de "w", uma vez que, ao abrir em "w", o ficheiro é apagado, enquanto que, abrindo em "a", o texto é adicionado (*appended*). Mais tarde, apercebemo-nos que, antes de escrever os dados, temos de limpar o ficheiro *report.csv* de modo a que o ficheiro tenha apenas os dados do último teste que foi feito. Assim, antes de iniciar a escrita do ficheiro, abrimo-lo em modo "wb", o que faz com que o conteúdo do ficheiro seja apagado antes de se começar a escrever.

2.5 Robustez

No que toca às exceções, decidimos verificar apenas se ocorreu *timeout* ou se houve um *socket error*. Assim, sempre que enviamos ou recebemos informação, testamos estas duas exceções. No que toca à existência do ficheiro *servers.json*, decidimos que, quando ocorresse um erro, o programa iria fechar. Esta é a única rota que faz sentido, uma vez que, se o ficheiro não existir, não será possível entrar em contacto com nenhum server e, assim, não faz sentido fazer o teste.

2.6 Encriptação

Para a encriptação, decidimos, após a escrita do *report.csv*, reler esse ficheiro e encriptá-lo linha a linha.

Capítulo 3

Resultados

Neste capítulo do relatório, decidimos fazer vários *speedtests*, usando várias fontes. Assim, será possível comparar os nossos resultados com os valores "reais".

Tabela 3.1: Resultado de speedtests de fontes diferentes

Fonte dos Dados	**Ping**	**Largura de Banda**
speedtest.net		
client.py		
fast.com	(não faz este teste)	
legacy.speedtest.net		

Capítulo 4

Análise

Analisa os resultados.

Capítulo 5

Conclusões

Apresenta conclusões.

Contribuições dos autores

Resumir aqui o que cada autor fez no trabalho. Usar abreviaturas para identificar os autores, por exemplo AS para António Silva. No fim indicar a percentagem de contribuição de cada autor.

Acrónimos