Trabalho de aprofundamento 2

Objetivos:

- Planeamento e preparação do trabalho de aprofundamento 2.

Este guião apresenta as regras o segundo trabalho de aprofundamento. Os exercícios sugeridos dão os primeiros passos para a concretização do trabalho recorrendo às ferramentas estudadas.

1.1 Regras

O trabalho deve ser realizado por um grupo de 2 alunos e entregue, via a plataforma `https://elearning.ua.pt`, dentro do prazo lá indicado. A entrega deverá ser feita por **apenas um dos membros** do grupo e deve consistir de **um único arquivo** `.zip`, `.tgz` (TAR comprimido por `gzip`) ou `.tbz` (TAR comprimido por `bzip2`).

O arquivo deve conter o código desenvolvido assim como o ficheiro PDF final do relatório, todos ficheiros com código fonte (`.tex`, `.bib`, etc.) e todas as imagens ou outros recursos necessários à compilação do código ou documento.

Na elaboração do relatório recomenda-se a adopção do estilo e estrutura de relatório descrito nas aulas teórico-práticas e a utilização de recursos de escrita como: referências a fontes externas, referências a figuras e tabelas, tabela de conteúdo, resumo, conclusões, etc. O objectivo do relatório é descrever a motivação, a implementação (não é só o código, mas também o algoritmo), apresentar testes que comprovem o seu funcionamento correto e analisar os resultados obtidos.

É obrigatório incluir uma secção "Contribuições dos autores" onde se descrevem resumidamente as contribuições de cada elemento do grupo e se avalia a percentagem de trabalho de cada um. Esta auto-avaliação poderá afetar a ponderação da nota a atribuir a cada elemento.

1.2 Avaliação

A avaliação irá incidir sobre:

1. **cumprimento dos requisitos apresentados**,
2. **qualidade do código produzido e comentários**,
3. **testes unitários e funcionais realizados**,
4. **estrutura e conteúdo do relatório**,
5. **utilização das funcionalidades de tarefas do code.ua e git**.
6. **o suporte de segurança adicionado**

Relatórios meramente descritivos sem qualquer descrição da aplicação, apresentação dos resultados obtidos, testes efetuados, ou discussão serão fracamente avaliados.

Só serão avaliados trabalhos enviados via a plataforma `https://elearning.ua.pt`. Ficheiros corrompidos ou inválidos não serão avaliados à posteriori e não será permitido o reenvio.

Deve ser utilizado um projeto na plataforma Code.UA, com um identificador segundo o formato `labi2019-ap2-gX`. **Substitua o caratere X pelo número 1. Se não for possível criar este projeto, incremente o número até que ele seja aceite. Não use valores aleatórios.**

1.3 Tema Proposto

Um aspeto constante no acesso à internet prende-se com o cumprimento da velocidade anunciada no contrato do serviço. É comum a existência de acessos à Internet com velocidades de várias centenas de megabits, especialmente nos centros urbanos, mas nem sempre estes valores estão realmente disponíveis. Para os clientes é importante determinar se o serviço contratado corresponde ao oferecido.

De forma a determinar a largura de banda disponível para obtenção de conteúdos, existe uma rede de servidores que permitem aos clientes avaliar a qualidade da ligação, especialmente latência e largura de banda de download e upload.

O objetivo deste trabalho é criar um cliente para estes servidores, nomeadamente os pertencentes ao serviço `speedtest`. O cliente deverá efetuar testes periódicos usando

múltiplos servidores. Ao longo da sua execução, deverá emitir um relatório, em formato Comma Separated Values (CSV)[1], contendo os dados obtidos (`report.csv`).

A lista de servidores é fornecida pelos docentes, consistindo num ficheiro no formato JavaScript Object Notation (JSON)[2].

O cliente possui os seguintes requisitos funcionais:

1. Se for iniciado com um número de argumentos inferior a 3, este deverá imprimir a ajuda no seguinte formato: `python3 client.py interval num [country or id]`

2. Se for iniciado com um qualquer argumento inválido (tipo errado, valor incorreto ou não encontrado), deverá ser apresentada uma mensagem de erro respetiva ao argumento que gera o erro.

3. Se for indicado um texto no terceiro argumento, o teste deverá ser realizado para um qualquer servidor aleatório daquele país.

4. Se for indicado um valor inteiro no terceiro argumento, o teste deverá ser realizado para o servidor com esse identificador.

5. O primeiro argumento deverá consistir num valor inteiro positivo, especificando o tempo que decorre entre dois testes realizados.

6. O segundo argumento deverá consistir num valor inteiro positivo, especificando o número de testes a serem realizados.

7. Caso não seja possível contactar um servidor específico, o programa deve apresentar uma mensagem significativa e iniciar um novo teste após o intervalo especificado. Neste caso, a largura de banda registada deverá será de 0 e a latência será de -1.

8. O ficheiro `report.csv` deverá possuir a seguinte estrutura: `contador, id do servidor, data e hora no formato ISO, latência, largura de banda, check`.

9. O campo `check` deverá conter a síntese calculada sobre a concatenhação de todos os campos anteriores, pela ordem apresentada, e sem qualquer separador.

10. O cliente deverá realizar uma descarga entre 10MB e 100MB, ou até que passem 10 segundos.

11. A taxa de largura de banda deve ser calculada pelo número de octetos recebidos sobre o tempo decorrido, após ter sido obtido 1MB.

12. A latência deve ser calculada pelo tempo médio de 10 transações `PING/PONG`.

13. O cliente deverá ler um ficheiro denominado `key.priv`, contendo uma chave privada RSA.

14. Ao terminar o cliente deverá escrever um ficheiro `report.sig` com uma assinatura do relatório por esta chave.

1.4 Protocolo utilizado

O protocolo utilizado pelos servidores opera sobre ligações Transmission Control Protocol (TCP)[3], efetuadas para o endereço e porta especificadas no ficheiro `servers.json`. As mensagens trocadas são simples mensagens de texto terminadas pelo caratere `\n`.

Consideram como relevantes os seguintes comandos enviados pelos clientes:

1. `HI\n`: Permite identificar a versão de software, a data e a hora do servidor. Uma resposta típica será `HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21`.

2. `PING <timestamp>\n`: Permite enviar ao servidor um valor inteiro (`timestamp`) com o tempo atual em milisegundos, sendo que o servidor irá responder com uma mensagem `PONG <timestamp>\n`, com o tempo do servidor. Analizando o tempo decorrido no envio consecutivo de várias mensagens (ex, 10) para o servidor, e respetiva resposta, permite-se determinar a latência da comunicação.

3. `DOWNLOAD <value>\n`: Permite indicar que pretendemos receber uma quantidade específica de dados (`value`) em octetos. O servidor irá imediatamente iniciar o envio de dados aleatórios com a dimensão pedida. O tempo decorrido pela receção destes dados permitirá determinar a largura de banda disponível com o servidor.

4. `QUIT\n`: Termina a sessão de testes.

As mensagens trocadas numa sessão para testar o protocolo seriam as seguintes:

```
HI\n
HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21\n
PING 1553111020189\n
PONG 1553111018911\n
PING 1553111020201\n
PONG 1553111018923\n
PING 1553111020214\n
PONG 1553111018936\n
PING 1553111020226\n
PONG 1553111018948\n
```

```
PING 1553111020238\n
PONG 1553111018960\n
DOWNLOAD 50\n
DOWNLOAD 7$@?8T|~@NP.p[g6cG|aGjA@z#laL}C:u=]hb ?\n
QUIT\n
```

1.5 Notas importantes

O módulo `time` permite obter o instante de tempo atual em microsegundos (`time.time()`) e realizar pausas no programa (`time.sleep(int)`). A diferença de tempo entre dois pontos da execução pode ser obtida com chamadas repetidas ao módulo `time`.

O módulo `datetime` permite obter a data e horas atuais (`datetime.datetime()`)

Um argumento corresponde a um valor inteiro se todos os seus carateres forem dígitos (`isdigit()`)

O módulo `json` permite importar rapidamente o ficheiro `servers.json`

Deve ser utilizado o módulo `csv` para acrescentar dados ao ficheiro.

O método `flush()` de um ficheiro pode ser utilizado para forçar a escrita dos dados para um ficheiro, mesmo antes do fecho do mesmo.

Importante: A determinação das caraterísticas do serviço de Internet é mais complexa do que a implementação resultante deste cliente. A largura de banda depende da localização do servidor, da hora do dia, da ligação entre o computador cliente e o quipamento que fornece ligação ao local, do código em sí, da atividade de clientes vizinhos, etc... . O protocolo TCP também possui uma característica denominada *slow start*, que requer um processamento ligeiramente mais cuidado. Embora forneça valores indicativos, este cliente não deve ser utilizado para a determinação da qualidade fiável de uma ligação ou para iniciar processos de reclamação junto dos operadores.

Glossário

CSV Comma Separated Values

JSON JavaScript Object Notation

TCP Transmission Control Protocol

Referências

[1] Y. Shafranovich, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, RFC 4180 (Informational), Internet Engineering Task Force, out. de 2005.

[2] E. T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 7159, Internet Engineering Task Force, mar. de 2014.

[3] J. Postel, *Transmission Control Protocol*, RFC 793 (Standard), Updated by RFCs 1122, 3168, 6093, Internet Engineering Task Force, set. de 1981.