

Trabalho de aprofundamento

UA

João Gameiro, Pedro Pereira



VERSAO

Trabalho de aprofundamento

DETI
UA

João Gameiro, Pedro Pereira
(93097) joao.gameiro@ua.pt, (93196) pedrocjdpereira@ua.pt

23-04-2018

Resumo

Este relatório tem como objetivo a descrição e apresentação dos resultados do código desenvolvido no âmbito do projeto Trabalho de Aprofundamento 2 (AP2) da disciplina de Laboratórios de Informática (LABI) do curso Mestrado Integrado em Engenharia de Computadores e Telemática (MIECT).

De maneira a que um cliente possa verificar se a velocidade da sua internet corresponde à velocidade estabelecida previamente em contrato, existe uma rede de servidores, em que é possível avaliar a qualidade da ligação. O objetivo deste trabalho era criar um cliente para esses servidores, para que posteriormente pudesse ser feita uma avaliação da velocidade da internet.

Assim sendo, serão apresentados todos os recursos usados neste trabalho, desde biografias até módulos importados no código. O código desenvolvido será interpretado, tal como os seus resultados e é esperado que no fim da leitura deste relatório todo o código e resultados do mesmo sejam mais claros.

Quanto ao seu conteúdo, este relatório começa por descrever o código de programa, desde o modo como verifica os argumentos de entrada, ao estabelecimento de conexão com o servidor, passa depois pela maneira como são efetuados os cálculos da latência e largura de banda e como os seus resultados são escritos num ficheiro csv. De seguida são efetuados testes para provar o funcionamento do código. Quando são inseridos erros propositados o programa reage corretamente, terminado e apresentando a mensagem de erro apropriada e, na ausência de erros, o programa funciona sem problemas e o ficheiro csv apresenta toda a informação correta. Para acabar o trabalho apresentam-se conclusões sobre o mesmo.

Conteúdo

1	Introdução	1
2	Metodologia	2
2.1	Módulos Importados	2
2.2	Condições iniciais	2
2.2.1	Argumentos de entrada	2
2.2.2	Intervalo e Número de Testes a Realizar	3
2.2.3	País ou id	3
2.2.4	Host	3
2.3	Comunicação com o servidor	4
2.3.1	HI	5
2.3.2	PING	5
2.3.3	DOWNLOAD	6
2.3.4	QUIT	7
2.4	Latência e Largura	7
2.5	Escrita dos Resultados	8
2.6	Chave Privada	8
3	Resultados e Análise	9
3.1	Erros Iniciais Propositados	9
3.2	Testes Funcionais com ID e com País	9
3.3	Erro DOWNLOAD	10
4	Conclusões	13
4.1	Code UA e ShareLatex	14
4.2	Nota Final	14

Capítulo 1

Introdução

O objetivo do programa é criar um cliente para servidores que servem para avaliação da velocidade da internet e a qualidade da ligação, permitindo ao cliente verificar se as capacidades da sua internet correspondem às estabelecidas previamente em contrato.

Este documento está dividido em quatro capítulos. Depois desta introdução, no Capítulo 2 é apresentada a metodologia seguida na elaboração do código. No Capítulo 3 são apresentados os resultados obtidos através de testes efetuados no programa, e estes mesmos resultados são discutidos e analisados. Finalmente, no Capítulo 4 são apresentadas as conclusões do trabalho.

Capítulo 2

Metodologia

Neste capítulo será descrito todo o código desenvolvido, indicando e explicando a funcionalidade que o mesmo implementa e a maneira como a mesma foi implementada.

2.1 Módulos Importados

Tabela 2.1: Tabela Módulos Importados e as suas respectivas utilidades

Módulo	Utilidade
sys	Possibilita a passagem de argumentos ao programa
time	Determinação do tempo atual e para paragem do mesmo
datetime	Determinar a data atual
json	Abertura do ficheiro servers.json
csv	Escrita de dados num ficheiro .csv
socket	Para permitir a comunicação com o servidor
random	Calcular um valor aleatório
RSA	Para síntese de uma chave privada
PKCS1_OAEP	útil na sínteses do ficheiro report.sig

2.2 Condições iniciais

2.2.1 Argumentos de entrada

Foi criado um if, para que se o número dos argumentos de entrada for menor que 3, o programa irá imprimir uma mensagem de erro a indicar esse acontecimento e posteriormente outra mensagem a referir o método de utilização (Ex.: *Usage method: python3 client.py interval num [country or id]*).

Seguidamente fez-se uso do módulo sys para terminar o programa caso o número de argumentos não correspondesse ao pretendido.

2.2.2 Intervalo e Número de Testes a Realizar

Estas duas condições vão ser verificadas da mesma maneira.

Através do módulo sys vão ser lidos respectivamente o intervalo entre dois testes e o número de testes a serem realizados.

Sendo que são lidos dentro da instrução try, se ocorrer algum erro, a posterior instrução except irá "apanhá-lo" e imprimir uma mensagem de erro. Dentro do try foi também implementado um if em que se o valor inserido for menor ou igual a zero o programa irá terminar.

Nos dois caso foi usada a mesma metodologia.

2.2.3 País ou id

```
#pais ou id
try:
    if(sys.argv[3].isdigit()):
        id_s = sys.argv[3]
        x = 1
    else:
        country_s = sys.argv[3]
        x = 0
except:
    print("ERROR : id must be int and country must be string")
    sys.exit()
```

Figura 2.1: Código para testar o argumento de entrada nº3

Para o terceiro argumento foi novamente utilizado um try e dentro do mesmo testa-se se o argumento de entrada é composto por números ou letras. Dependendo do resultado é criada uma variável de estado para posterior utilização e se for por números considera-se como sendo o id (*id_s*), por letras como sendo o país para o qual deve ser realizado o teste (*country_s*).

Se ocorrer algum o erro o programa imprime uma mensagem a indicar o mesmo e termina.

2.2.4 Host

Para a obtenção do Host foi necessário a abertura e leitura do ficheiro servers.json, o que foi feito com a ajuda do módulo json.

ID

Com a variável de estado x criada anteriormente, foi possível agora obter o host a partir do *id_s* ou do *country_s*. Logo se x tomar o valor de 1, é criado um for que percorre todos os ids pertencentes ao ficheiro servers.json e quando encontrar o correspondente para. Depois esse valor é guardado na variável *final_id* para escrita no ficheiro report.csv e o host correspondente a esse id é obtido e guardado também em variáveis (utilização do método split) para fazer a ligação ao servidor.

Country

Se x tomar o valor 0, os países presentes no ficheiro json vão ser corridos e quando se encontrar o correspondente ao inserido no argumento é adicionado a uma lista l o seu respetivo host. Adicionalmente é criada outra lista i para armazenar os ids. Após isto faz-se uso do módulo random para escolher da lista um host para usar na conexão ao servidor. É também criado um ciclo for para percorrer todos os hosts e obter a posição do host escolhido que depois será usada para alcançar o valor de *final_id*.

```
if x==1:
    for serv in info['servers']:
        if(int(id_s) == serv['id']):
            break

        final_id = serv['id']

        l0 = []
        l0 = (serv['host'].split(':'))
        b_data = l0[0]
        addr = l0[1]
        #obter o host a partir do country
else:
    for serv in info['servers']:
        if(country_s == serv['country']):
            l.append(serv['host'])
            i.append(serv['id'])

    if(len(l) == 0):
        print("ERROR: Country does not exist")
        sys.exit()

    l1 = []
    l1 = (random.choice(l).split(':'))
    b_data = l1[0]
    addr = l1[1]

    for n in range(0, len(l)):
        if(str(l1) == l[n]):
            break

    final_id = i[n]
```

Figura 2.2: Código para obtenção do host

Após tudo isto, com apoio do módulo csv é aberto o ficheiro report.csv para a escrita dos dados que vão ser obtidos.

2.3 Comunicação com o servidor

Toda a comunicação com o servidor é feita dentro de um ciclo for que se repete n vezes (sendo o n o número de testes inserido como argumento).

Para a comunicação com o servidor vão ser usadas 4 Strings:

HI - que vai receber a resposta HELLO com identificação do software;

PING - acompanhado com `time.time()` que corresponde ao tempo em microsegundos que irá receber a resposta PONG;

DOWNLOAD - acompanhado pela quantidade em octetos que o utilizador quiser.

QUIT - para terminar o programa

Assim sendo, dentro de um try é feita a conexão ao servidor criando-se um socket que terá como argumentos *b_data* e *addr* obtidos anteriormente. Se a ligação for bem sucedida então é apresentada uma mensagem a prová-lo e é registado o tempo. Caso contrário é apresentada uma mensagem de erro e a largura e a latência tomam valores 0 e -1 respectivamente pois estão na presença de uma conexão falhada.

```
#criação do socket que irá possibilitar a comunicação com o servidor
try:
    tcp_s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    tcp_s.connect((str(b_data), int(addr)))
    print("Connected to : " + str(b_data) + " " + str(addr))
    inicial_time = time.time() #registo do tempo a partir do momento em que é aberto o servidor
except:
    print("ERROR: Connection Failed")
    largura = 0
    latencia = -1
```

Figura 2.3: Ligação ao servidor

2.3.1 HI

Das strings criadas anteriormente é enviada ao servido a string HI. Em seguida é criada uma variável *resulthello* que vai receber a resposta do servidor e que será impressa (será a identificação do software).

```
#HI\n
tcp_s.send(hello.encode())
resulthello = tcp_s.recv(4096)
print("HI")
print(str(resulthello))
```

Figura 2.4: Comando HI

2.3.2 PING

Antes do envio do comando PING para o servidor é criada uma lista que irá armazenar o valor devolvido pelo PONG.

Como em cada teste ocorrem 10 interações PING/PONG, foi criado um ciclo for que irá repetir-se 10 vezes.

Dentro de cada ciclo é em primeiro lugar enviado para o servidor o comando PING acompanhado com o tempo atual em microsegundos, em segundo lugar é

armazenada a resposta devolvida (PONG + tempo do servidor) numa variável. Essa variável é dividida (método split) em PONG e tempo do servidor. Da divisão retiramos o tempo do servidor e adiciona-mo-lo à lista criada anteriormente.

Após todos os 10 ciclos completos é impresso no ecrã o conteúdo da lista com o número à frente que serve como contagem do número de PONGS recebidos (o último tem de ser sempre 10).

```
#PING
#criação de uma lista para armazenar o resultado dos pongs
listpong = []
#obtenção dos pongs
for z in range(0, 10):
    tcp_s.send(ping.encode())
    resultping = tcp_s.recv(4096)
    pong = str(resultping).split(' ')
    pong0 = (pong[1])
    listpong.append(pong0[0 : len(pong0)-3])
z=1
for i in range(0, len(listpong)):
    print(("PONG "+str(listpong[i]))+" "+str(z))
    z = z + 1
```

Figura 2.5: Comando PING

2.3.3 DOWNLOAD

Antes da início do download é registado o tempo, que servirá para verificar se já passaram 10 segundos ou não. Por isso todo o processo de troca de informação envolvendo o DOWNLOAD é realizado dentro de um if que certifica que o DOWNLOAD tem de ser feito até que passem 10 segundos.

É iniciada uma variável *numd* que é recebida pelo teclado e que tem de pertencer ao intervalo [10, 100] (daí o assert presente na linha seguinte).

Após isso é registado o tempo antes do início do download e enviado para o servidor o valor inserido através do comando DOWNLOAD. A seguir à recepção da resposta do servidor regista-se o tempo (irá ser usado para calcular a largura de banda) e imprime-se a informação recebida.

```
#DOWNLOAD
down_time = time.time()
if down_time - inicial_time < 10: #até 10 segundos
    numd = input("DOWNLOAD ") #o cliente insere o valor pretendido
    assert (int(numd) >= 10 and int(numd) <= 100), "ERROR: DOWNLOAD belongs to [10, 100]"
    download = str(download + str(int(numd)) + "\n")
    begin_download = time.time()
    tcp_s.send(download.encode()) # é enviado para o servidor
    resultdownload = tcp_s.recv(4096)
    end_download = time.time() #regista o fim do download
    print(resultdownload)
```

Figura 2.6: Comando DOWNLOAD

2.3.4 QUIT

Após termos todos os dados necessários para o objetivo pretendido, é enviado ao servidor o comando QUIT o que irá fechar a ligação com o mesmo. É fechado também o socket com o comando `tcp_s.close()`.

```
#QUIT
print(quit)
tcp_s.send(quit.encode())
tcp_s.close()
```

Figura 2.7: Comando QUIT

2.4 Latência e Largura

- **Latência;**

Ao calcular a média dos valores retornados pelos PONGS obtemos a latência. Por isso foi criada uma variável `soma` que corresponde à soma e todos os valores da lista que armazenava os PONGS, que foi dividida pelo número de elementos dessa lista.

O resultado foi imprimido no terminal.

- **Largura de Banda;**

A Largura de Banda foi calculada ao ser dividido o valor do download por um delta tempo. Esse delta tempo corresponde ao tempo que o servidor demorou a enviar a quantidade indicada anteriormente e foi calculado com a ajuda dos valores de `end_download` e `begin_download`.

Tal como a latência, também o valor da largura foi posteriormente imprimido.

```
#cálculo da latência
soma = 0
for z in range(0, len(listpong)):
    soma = soma + float(listpong[z])
latencia = soma/len(listpong)
print("Latência : " + str(latencia))

#cálculo da largura de banda
largura = (float(numd) / (end_download - begin_download))
print("Largura de Banda : " + str(largura) + "\n")
```

Figura 2.8: Cálculo da Latência e da Largura de Banda

2.5 Escrita dos Resultados

Os resultados obtidos, da latência e da largura de banda, tal como o id do servidor foram escritos num ficheiro report.csv.

Ficheiro esse que contém também a data actual (uso do módulo datetime), o número de testes e um campo check que contém todos os dados juntos sem qualquer tipo de separador.

De referir mais um facto, no fim da escrita dos dados, existe um trecho que para o tempo se houver mais algum teste a ser feito, de acordo com o valor especificado nos argumentos de entrada, ou seja é o intervalo de espera entre cada teste.

2.6 Chave Privada

Após todas as informações foi criada uma chave privada *key* e lido um ficheiro key.priv que continha síntese dessa chave. Foi também criado o report.sig que contém as mesmas informações que o report.csv e que tem a assinatura da chave privada.

```
#Criação de chaves RSA e Leitura do ficheiro que as contem key.priv
key = RSA.generate(1024)
filef = open("key.priv", "wb")
k = key.exportKey("PEM", "senha")
filef.write(k)
filef.close()

filefi = open("key.priv", "rb")
k = RSA.importKey(filefi.read(), "senha")
filefi.close()

#Ficheiro report.sig com assinatura da chave
cipher = PKCS1_OAEP.new( k )
with open("report.sig", "wb") as file:
    with open("report.csv", "r") as ori:
        for line in ori.read():
            file.write(cipher.encrypt(line.encode("utf-8")))
```

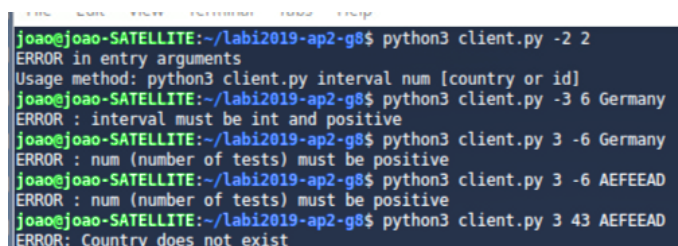
Figura 2.9: Chave privada e ficheiro report.sig

Capítulo 3

Resultados e Análise

Neste capítulo vão ser descritos os resultados obtidos pelo programa.

3.1 Erros Iniciais Propositados



```
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py -2 2
ERROR in entry arguments
Usage method: python3 client.py interval num [country or id]
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py -3 6 Germany
ERROR : interval must be int and positive
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py 3 -6 Germany
ERROR : num (number of tests) must be positive
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py 3 -6 AEFEEAD
ERROR : num (number of tests) must be positive
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py 3 43 AEFEEAD
ERROR: Country does not exist
```

Figura 3.1: Erros iniciais

Como podemos ver na Figura 3.1, se for iniciado com argumento inferior a 3 imprime uma mensagem de erro e termina, tal como se o intervalo e o número de testes for menor que 0. Se o país não existir também não inicia.

Se for indicado um id inválido, o programa vai por opção de default realizar os testes para o host *speedtest.ushuaiavision.com.ar 8080*

3.2 Testes Funcionais com ID e com País

Ao realizarmos o teste para o id 21239 verificamos que corre tudo como o esperado(são realizados dois testes, 1 segundo de espera entre cada um e todas as funcionalidades descritas comportam-se como esperado)(Figura 3.2).

Se realizarmos para um país constamos que também ocorre tudo como esperado.(Figura 3.3)

O report.csv contém todas as informações especificadas nos testes e foi criada uma chave e um ficheiro report.sig.

```
File Edit View Terminal Tabs Help
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py 1 2 21239
Teste 1

Connected to : speedtest.oppinord.no 8080
HI
b'HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21\n'
PONG 1556205682777 1
PONG 1556205682879 2
PONG 1556205682981 3
PONG 1556205684493 4
PONG 1556205684595 5
PONG 1556205690390 6
PONG 1556205690493 7
PONG 1556205690600 8
PONG 1556205690702 9
PONG 1556205690804 10
DOWNLOAD 55
b'DOWNLOAD a6R\x7f4@lqj=M;) {_. gyU\^\`#<4++GY*k[|l16qs.-bozo\n'
QUIT

Latência : 1556205687071.4
Largura de Banda : 515.2730635383674

Teste 2

Connected to : speedtest.oppinord.no 8080
HI
b'HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21\n'
PONG 1556205693848 1
PONG 1556205693949 2
PONG 1556205694050 3
PONG 1556205694153 4
PONG 1556205694254 5
PONG 1556205694355 6
PONG 1556205694457 7
PONG 1556205694559 8
PONG 1556205694660 9
PONG 1556205694762 10
DOWNLOAD 22
b'DOWNLOAD a6R\x7f4@lqj=M;\n'
QUIT

Latência : 1556205694304.7
Largura de Banda : 216.15160530150058

joao@joao-SATELLITE:~/labi2019-ap2-g8$
```

Figura 3.2: Teste Funcional ID

3.3 Erro DOWNLOAD

Na Figura 3.4 é possível verificar que se o valor do DOWNLOAD não pertencer a [10, 100] ocorre um erro.

Na Figura 3.5 também é possível ver a chave RSA e um ficheiro report.csv de um teste anteriormente realizado.

```
File Edit View Terminal Tabs Help
joao@joao-SATELLITE:~/labi2019-ap2-g8$ python3 client.py 2 2 Portugal
Teste 1

Connected to : porto.speedtest.net.zon.pt 8080
HI
b'HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21\n'
PONG 1556206467120 1
PONG 1556206467129 2
PONG 1556206467190 3
PONG 1556206467205 4
PONG 1556206467219 5
PONG 1556206467228 6
PONG 1556206467245 7
PONG 1556206467263 8
PONG 1556206467273 9
PONG 1556206467280 10
DOWNLOAD 15
b'DOWNLOAD +ZzZU\n'
QUIT

Latência : 1556206467215.2
Largura de Banda : 1511.315669365106

Teste 2

Connected to : porto.speedtest.net.zon.pt 8080
HI
b'HELLO 2.6 (2.6.9) 2019-02-20.2246.62a8e21\n'
PONG 1556206471675 1
PONG 1556206471682 2
PONG 1556206471689 3
PONG 1556206471696 4
PONG 1556206471702 5
PONG 1556206471709 6
PONG 1556206471715 7
PONG 1556206471721 8
PONG 1556206471727 9
PONG 1556206471733 10
DOWNLOAD 99
b'DOWNLOAD +ZzZUTh`rX?5N0Gv\`x7f{\x7fud5+%yDrs9?,rdr}[Zo zw0QpesQf0f[,eOCA'[[bK6pk%ZnVvH!@5ds:0]\\87M1627p"
QUIT

Latência : 1556206471704.9
Largura de Banda : 19340.293246390313

joao@joao-SATELLITE:~/labi2019-ap2-g8$
```

Figura 3.3: Teste Funcional País

```
PONG 1556206657307 7
PONG 1556206657319 8
PONG 1556206657331 9
PONG 1556206657343 10
DOWNLOAD 1
Traceback (most recent call last):
  File "client.py", line 139, in <module>
    assert (int(numd) >= 10 and int(numd) <= 100), "ERROR: DOWNLOAD belongs to [
10, 100]"
AssertionError: ERROR: DOWNLOAD belongs to [10, 100]
joao@joao-SATELLITE:~/labi2019-ap2-g8$
```

Figura 3.4: Erro Download

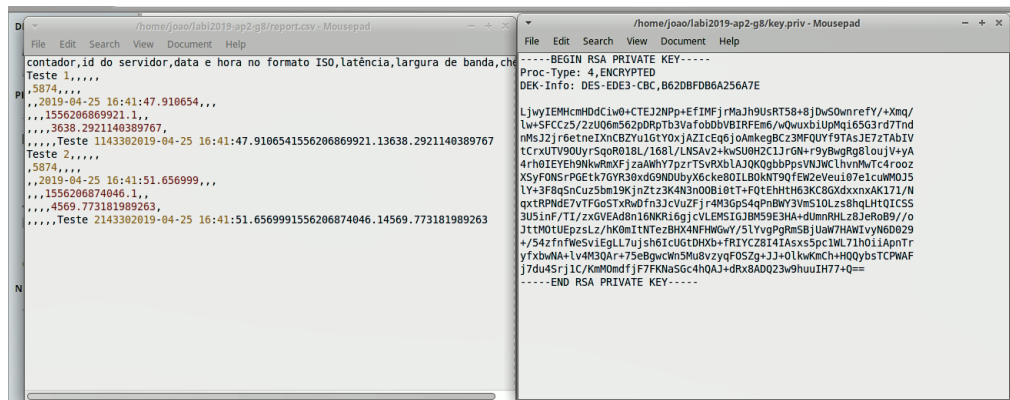


Figura 3.5: Chave RSA e report.csv

Capítulo 4

Conclusões

Com o trabalho concluído, é possível comprovar o correto funcionamento do programa criado, capaz de avaliar a velocidade da internet do cliente através da sua execução acompanhada por argumentos válidos. Compreendendo melhor o seu funcionamento entende-se, assim, a utilidade de uma aplicação como esta bem como a importância deste projeto para o desenvolvimento das capacidades de escrita de código em python dos seus autores.

Contribuições dos autores

Todo o código foi discutido e desenvolvido em conjunto. A estrutura do relatório foi também discutida em conjunto, sendo que João Gameiro (JG) escreveu os capítulos Capítulo 2 e Capítulo 3 e Pedro Pereira (PP) escreveu os capítulos Capítulo 1 e Capítulo 4. Considera-se que cada autor contribuiu igualmente para este trabalho logo a atribuição de percentagens é JG - 50% e PP - 50%.

4.1 Code UA e ShareLatex

Aqui encontra-se o link para o projeto usado na plataforma Code UA.

- <https://code.ua.pt/projects/labi2019-ap2-g8/repository>;

Para o desenvolvimento do relatório como foi usada a plataforma ShareLatex, por isso não existe nenhuma versão do mesmo no Code UA.

4.2 Nota Final

[1]

O código desenvolvido, conteúdo apresentado e estrutura do relatório tiveram como principal apoio os Guiões e PowerPoints da disciplina LABI da Universidade de Aveiro (UA) fornecidos aos alunos.

Acrónimos

AP2 Trabalho de Aprofundamento 2

LABI Laboratórios de Informática

UA Universidade de Aveiro

MIECT Mestrado Integrado em Engenharia de Computadores e Telemática

JG João Gameiro

PP Pedro Pereira

Bibliografia

- [1] D. do DETI que leccionam a disciplina de LABI (UA), *Guiões Laboratórios de Informática*, 2019. URL: <https://elearning.ua.pt/course/view.php?id=3470> (acedido em 25/04/2019).