TP4 - Access Links, Packet Loss & Duplication - Best Effort Internet
Grupo 4

Bruno Silva (a71385)
João Bernardo Freitas (a74814)
Eduardo Gil Rocha (a77048)

20 de Dezembro de 2019

Contents

Introdução

Este relatório tem como objetivo explicar o trabalho realizado em âmbito do quarto trabalho prático de Tecnologias e Protocolos Internet. Neste trabalho estudamos *Access Links* e os efeitos que a alteração de várias características destes, como o atraso e perca de pacotes, têm sobre a performance da rede.

1

Seguindo a especificação apresentada no enunciado, foi desenhada a seguinte topologia no emulador *CORE*:

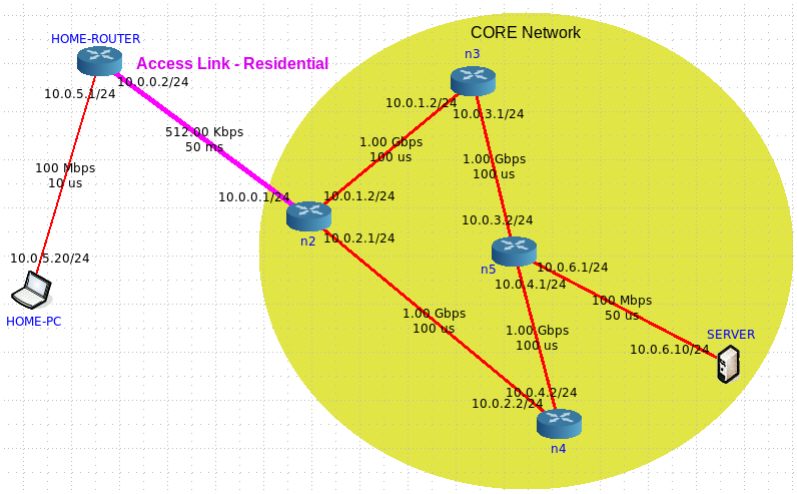

Figure 1: Topologia

1.1

De seguida iremos apresentar os testes de *ping* iniciais, que demonstram que há conetividade na rede. O primeiro teste foi feito a partir do nó n3 até ao nó n4.

```
root@n3:/tmp/pycore.43669/n3.conf# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_req=1 ttl=63 time=1.58 ms
64 bytes from 10.0.2.2: icmp_req=2 ttl=63 time=1.48 ms
64 bytes from 10.0.2.2: icmp_req=3 ttl=63 time=0.783 ms
64 bytes from 10.0.2.2: icmp_req=4 ttl=63 time=0.672 ms
64 bytes from 10.0.2.2: icmp_req=5 ttl=63 time=0.593 ms
```

Figure 2: Teste ping entre n3 e n4

O segundo teste foi entre o nó n2 e o servidor.

```
root@n2:/tmp/pycore.43669/n2.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_req=1 ttl=62 time=0.949 ms
64 bytes from 10.0.6.10: icmp_req=2 ttl=62 time=0.854 ms
64 bytes from 10.0.6.10: icmp_req=3 ttl=62 time=0.887 ms
64 bytes from 10.0.6.10: icmp_req=4 ttl=62 time=0.836 ms
64 bytes from 10.0.6.10: icmp_req=5 ttl=62 time=0.895 ms
```

Figure 3: Teste ping entre n2 e SERVER

Por fim, testou-se a conetividade entre os dois extremos da rede, ou seja, o servidor e o computador na rede residencial.

```
root@SERVER:/tmp/pycore.43669/SERVER.conf# ping 10.0.5.20
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data.
64 bytes from 10.0.5.20: icmp_req=1 ttl=60 time=126 ms
64 bytes from 10.0.5.20: icmp_req=2 ttl=60 time=104 ms
64 bytes from 10.0.5.20: icmp_req=3 ttl=60 time=124 ms
64 bytes from 10.0.5.20: icmp_req=4 ttl=60 time=102 ms
64 bytes from 10.0.5.20: icmp_req=5 ttl=60 time=143 ms
```

Figure 4: Teste ping entre HOME-PC e SERVER

Para a segunda etapa de testes de conetividade, realizamos testes *ping* entre os nós HOME-PC e SERVER, usando desta vez duas *flags*: *-c* e *-s*. A primeira é usada para definir o número máximo de pacotes a enviar e a segunda para definir o tamanho de cada pacote.

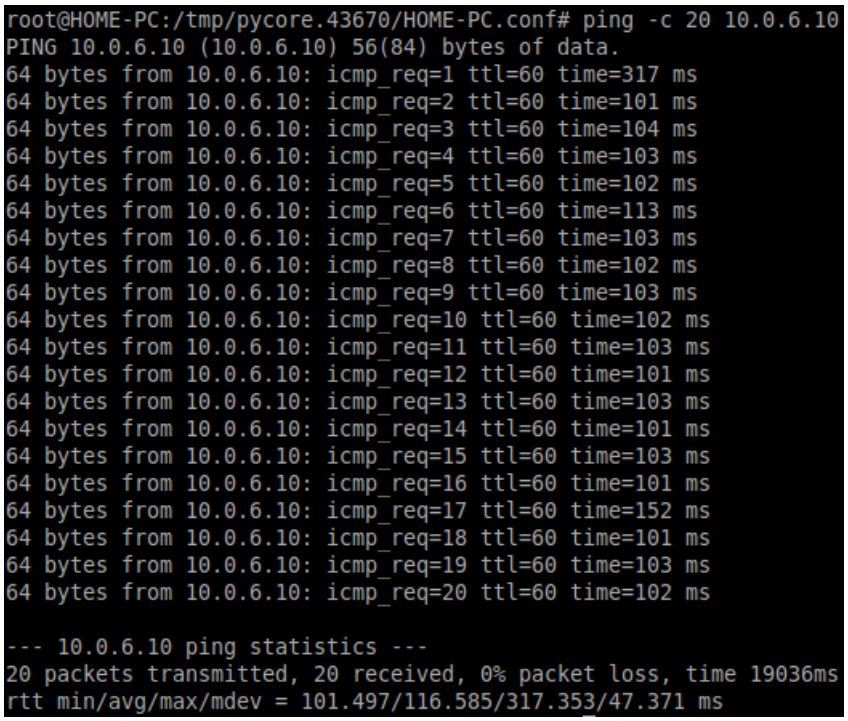

Figure 5: Teste ping entre HOME-PC e SERVER com número de pacotes

Nestes testes, pode-se ver que as mensagens tiveram um *round trip time* médio de 116.585 ms, com um atraso mínimo de 101.497 ms.

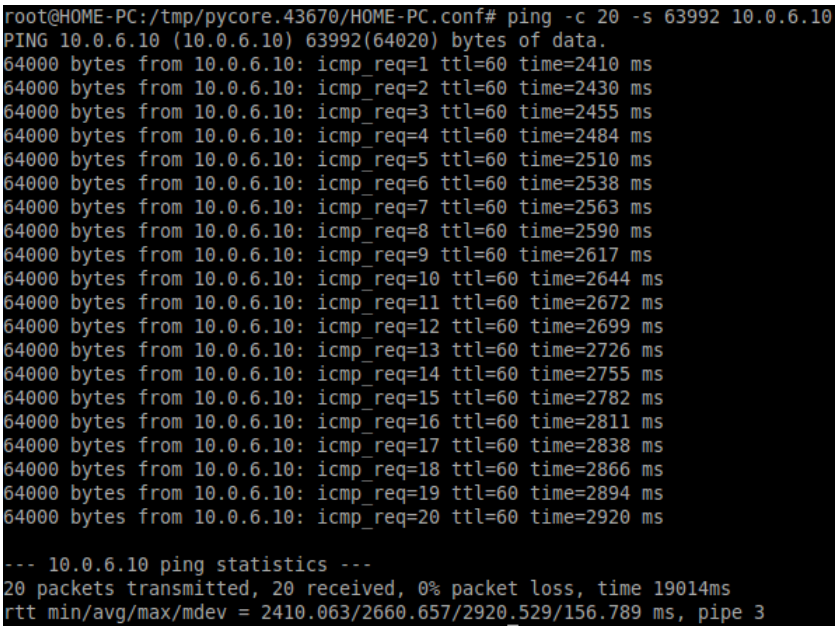

Figure 6: Teste ping entre HOME-PC e SERVER com número e tamanho de pacotes

Com o aumento do tamanho dos pacotes para 64KB, vemos que o tempo médio sobe para 2660.657 ms. Este resultado é esperado visto que um pacote de tamanho 64KB, ou seja, de 512Kb, demorará um segundo a passar pela ligação residencial. Portanto podemos esperar que o *round trip time* seja maior do que dois segundos. Podemos concluir a partir disto que o fato de que o *Access Link* residencial está limitado a 512Kbps introduz um *bottleneck* significativo à rede.

1.2

```
root@HOME-PC:/tmp/pycore.54340/HOME-PC.conf# ping -c 20 -s 63992 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 63992(64020) bytes of data.
64000 bytes from 10.0.6.10: icmp_req=1 ttl=60 time=105 ms
64000 bytes from 10.0.6.10: icmp_req=2 ttl=60 time=109 ms
64000 bytes from 10.0.6.10: icmp_req=3 ttl=60 time=116 ms
64000 bytes from 10.0.6.10: icmp_req=4 ttl=60 time=113 ms
64000 bytes from 10.0.6.10: icmp_req=5 ttl=60 time=111 ms
64000 bytes from 10.0.6.10: icmp_req=6 ttl=60 time=113 ms
64000 bytes from 10.0.6.10: icmp_req=7 ttl=60 time=117 ms
64000 bytes from 10.0.6.10: icmp_req=8 ttl=60 time=111 ms
64000 bytes from 10.0.6.10: icmp_req=9 ttl=60 time=120 ms
64000 bytes from 10.0.6.10: icmp_req=10 ttl=60 time=118 ms
64000 bytes from 10.0.6.10: icmp_req=11 ttl=60 time=122 ms
64000 bytes from 10.0.6.10: icmp_req=12 ttl=60 time=121 ms
64000 bytes from 10.0.6.10: icmp_req=13 ttl=60 time=123 ms
64000 bytes from 10.0.6.10: icmp_req=14 ttl=60 time=123 ms
64000 bytes from 10.0.6.10: icmp_req=15 ttl=60 time=125 ms
64000 bytes from 10.0.6.10: icmp_req=16 ttl=60 time=121 ms
64000 bytes from 10.0.6.10: icmp_req=17 ttl=60 time=113 ms
64000 bytes from 10.0.6.10: icmp_req=18 ttl=60 time=108 ms
64000 bytes from 10.0.6.10: icmp_req=19 ttl=60 time=116 ms
64000 bytes from 10.0.6.10: icmp_req=20 ttl=60 time=110 ms

--- 10.0.6.10 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19046ms
rtt min/avg/max/mdev = 105.001/116.205/125.642/5.751 ms
```

Figure 7: Ping entre HOME-PC e SERVER com a bandwidth a 1Gps

Como podemos verificar a velocidade de transferência aumentou, de RTT médio 2660ms para 116ms, visto que o *bottleneck* na ligação **Access Link - Residential** foi eliminado. Porém a topologia continua a ter dois *bottlenecks* na ligação **HOME-PC**− >**HOME-ROUTER** e **n5**− >**Server**, logo não há vantagem em melhorar a ligação **Access Link -Residential** para lá dos 100Mbps.

Nota: Os delays na topologia também afectam adversamente a velocidade de transferência, visto que o delay total HOME-PC->SERVER é de 50,6 ms. Como é uma transferência em TCP esse valor é duplicado para 101,2ms.

1.3

Para proceder ao **file-transfer** para obter conclusões sobre o goodput entre o **HOME-PC** e **SERVER**, foi necessário criar um dummy file para teste. Para isso, foi gerado um ficheiro de tamanho 10**MB** com o commando **$ dd if=/dev/urandom of=file.txt bs=1048576 count=10**.

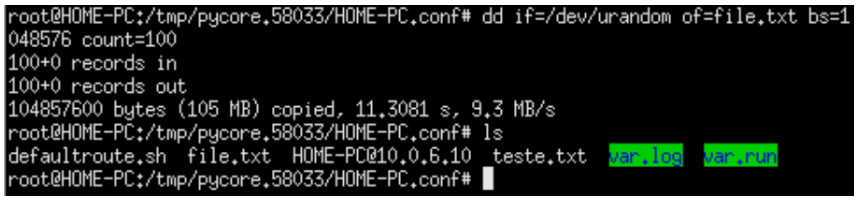

Figure 8: Geração do file.txt de tamanho 10MB

De seguida, com recurso ao comando **scp - secury copy**, que utiliza o protocolo **ssh**, transferiu-se o ficheiro do HOME-PC para o SERVER. Este processo foi feito algumas vezes para efeitos de uma maior amostra.

Figure 9: Transferência entre o HOME-PC e o SERVER

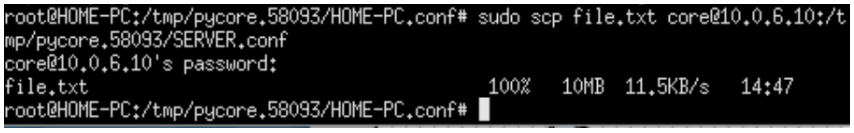

Figure 10: Transferência entre o HOME-PC e o SERVER

Figure 11: File.txt recebido no SERVER

Após a análise dos resultados, e sabendo que estamos perante **TCP** a nível da camada protocolar de transporte, conclui-se que apesar das elevada largura de banda nas ligações, o **delay** tem um grande impacto no tempo necessário na transferência do ficheiro. Se considerarmos que ao efetuar o traceroute do HOME-PC para o SERVER, demora cerca de 103**ms**, e isto para 60bytes, é normal que num ficheiro de dimensões muito superiores o tempo de espera seja muito mais significativo.
Além disso, como estamos a falar de **TCP**, é necessário ter em conta os **ACKs**, que associados aos delays estabelecidos nas ligações das topologias escalam ainda mais o tempo de espera; e a retransmissão de pacotes quando algo de errado acontece.
Por curiosidade, foi efetuado um **file-transfer** entre o HOME-PC e o SERVER, em que se removeu o delay da ligação **Access Link - Residential**.

Figure 12: Transferência do file.txt, sem delay na ligação residencial

2

2.1

2.1.1 Ping

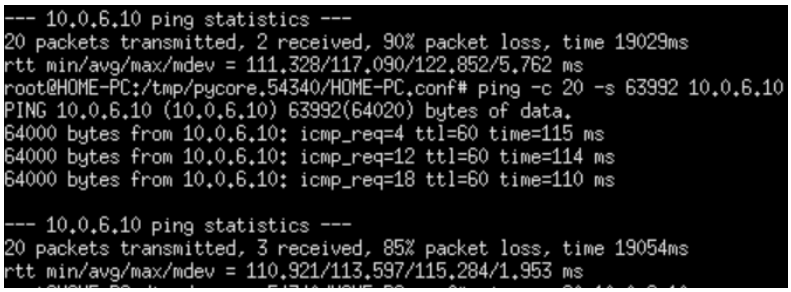

Figure 13: Ping entre HOME-PC e SERVER com ficheiro de 64Kbytes, com probabilidade de duplicados e perda a 2%

Como se pode observar, ao enviar pacotes grandes é provável que a perda de pacotes seja considerável visto que cada pacote é fragmentado e que basta um dos fragmentos não chegar ao destino para o pacote ser dado como perdido. Como tal decidimos fazer estes testes com pacotes de 64bytes.

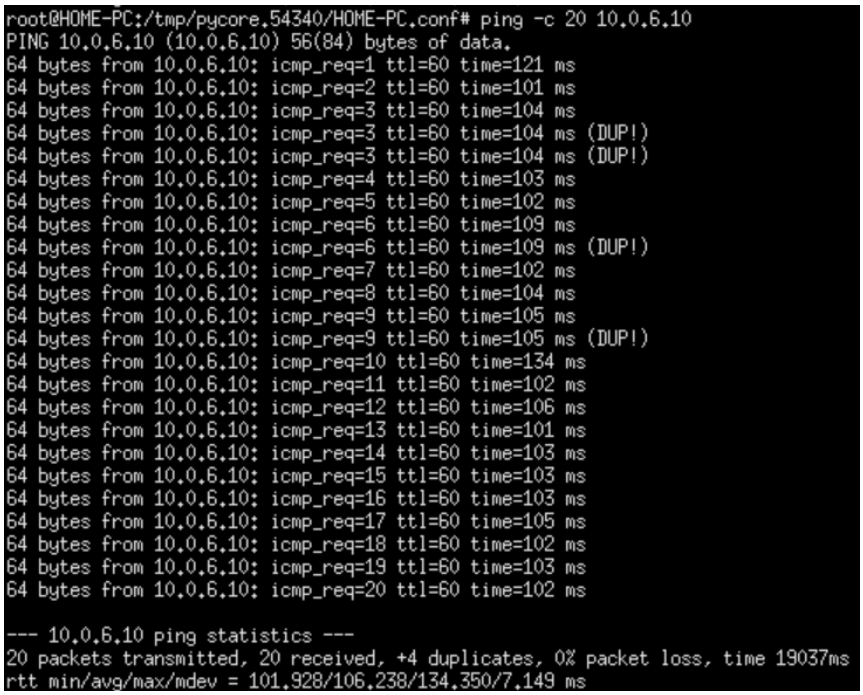

Figure 14: Ping entre HOME-PC e SERVER com ficheiro de 64bytes, com probabilidade de duplicados e perda a 2%

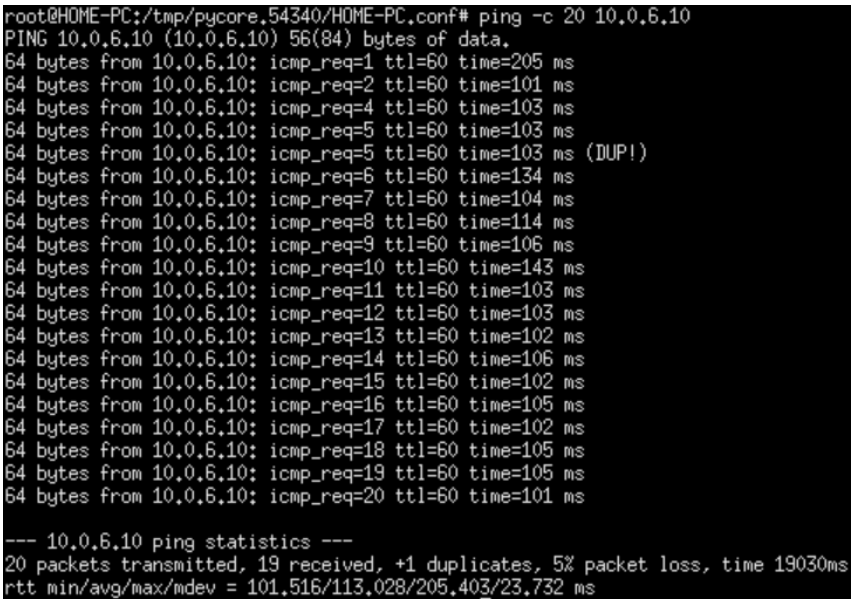

Figure 15: Ping entre HOME-PC e SERVER com ficheiro de 64bytes, com probabilidade de duplicados a 5% e perda a 2%

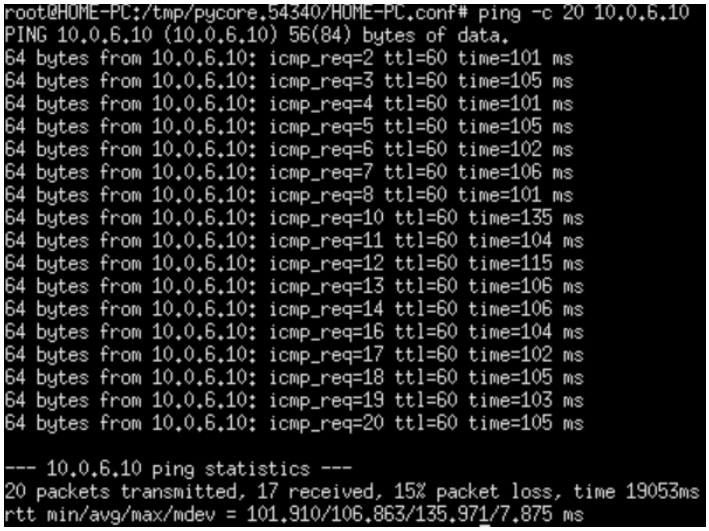

Figure 16: Ping entre HOME-PC e SERVER com ficheiro de 64bytes, com probabilidade de duplicados a 2% e perda a 10%

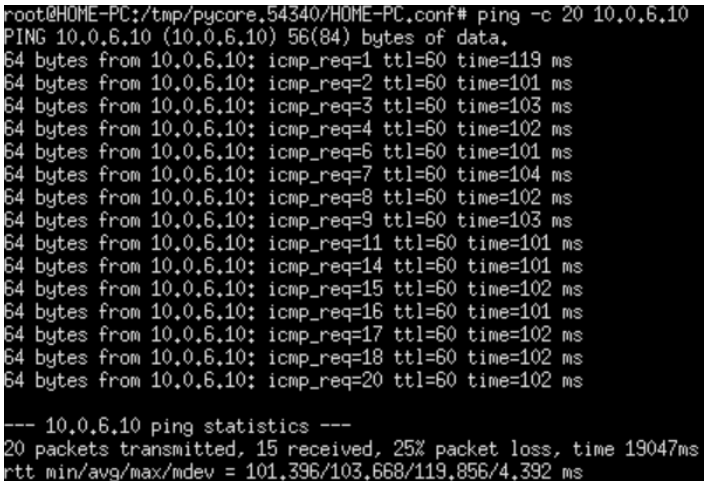

Figure 17: Ping entre HOME-PC e SERVER com ficheiro de 64bytes, com probabilidade de duplicados a 5% e perda a 10%

Nota: Se aumentássemos o nº de pacotes enviados para 100, o nº de pacotes perdidos/duplicados iria ser semelhante ás percentagens dadas.

2.1.2 File Transfer

Para os testes de file transfer decidimos utilizar **Secure Copy-scp** que opera sobre **TCP** e **SSH**. Criamos também um ficheiro de 5MB para transferir entre **HOME-PC** e **SERVER**, visto que é suficientemente grande para os resultados serem relevantes.

Figure 18: Criação do ficheiro teste.txt de 5MB

Para termos um *baseline* testamos a transferência sem pacotes duplicados ou perda de pacotes.

Figure 19: Transferência do ficheiro sem perda e duplicados

Figure 20: Transferência do ficheiro com 2% de perda e 2% duplicados

Figure 21: Transferência do ficheiro com 2% de perda e 5% duplicados

Figure 22: Transferência falhada do ficheiro com 10% de perda e 2% duplicados

Nota: Após 30 minutos desistimos deste teste visto que não progredia, como tal não vamos efectuar o teste com 10% de perda e 5% duplicados visto que iria ter o mesmo resultado

Como podemos ver, ao aumentar a percentagem de pacotes duplicados não há grande diferença em relação á velocidade de transferência, visto que o **HOME-PC** ao receber um **ACK** duplicado não é obrigado a reenviar pacotes, ao contrário do que acontece quando há pacotes perdidos, onde é obrigado a reenviar o pacote.

2.2

Para esta secção, aumentamos o valor do campo *delay* na configuração do *Access Link* residencial para dois segundos. Com esta nova configuração, realizamos testes com *ping* e *file transfer* entre o *HOME-PC* e o *SERVER*.

2.2.1 Ping

Para referência, estes seguintes *pings* foram feitos sem o aumento de atraso:

```
root@HOME-PC:/tmp/pycore.44883/HOME-PC.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_req=1 ttl=60 time=1.29 ms
64 bytes from 10.0.6.10: icmp_req=2 ttl=60 time=1.08 ms
64 bytes from 10.0.6.10: icmp_req=3 ttl=60 time=1.09 ms
64 bytes from 10.0.6.10: icmp_req=4 ttl=60 time=1.83 ms
64 bytes from 10.0.6.10: icmp_req=5 ttl=60 time=1.02 ms
```

Figure 23: Ping entre HOME-PC e SERVER

Com o aumento do atraso, verificamos os seguintes resultados:

```
root@HOME-PC:/tmp/pycore.44883/HOME-PC.conf# ping -c 20 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_req=1 ttl=60 time=4004 ms
64 bytes from 10.0.6.10: icmp_req=2 ttl=60 time=4079 ms
64 bytes from 10.0.6.10: icmp_req=3 ttl=60 time=4002 ms
64 bytes from 10.0.6.10: icmp_req=4 ttl=60 time=4023 ms
64 bytes from 10.0.6.10: icmp_req=5 ttl=60 time=4017 ms
64 bytes from 10.0.6.10: icmp_req=6 ttl=60 time=4003 ms
64 bytes from 10.0.6.10: icmp_req=7 ttl=60 time=4004 ms
64 bytes from 10.0.6.10: icmp_req=8 ttl=60 time=4027 ms
64 bytes from 10.0.6.10: icmp_req=9 ttl=60 time=5014 ms
64 bytes from 10.0.6.10: icmp_req=10 ttl=60 time=4005 ms
64 bytes from 10.0.6.10: icmp_req=11 ttl=60 time=4015 ms
64 bytes from 10.0.6.10: icmp_req=12 ttl=60 time=4004 ms
64 bytes from 10.0.6.10: icmp_req=13 ttl=60 time=4008 ms
64 bytes from 10.0.6.10: icmp_req=14 ttl=60 time=4017 ms
64 bytes from 10.0.6.10: icmp_req=15 ttl=60 time=4008 ms
64 bytes from 10.0.6.10: icmp_req=16 ttl=60 time=4013 ms
64 bytes from 10.0.6.10: icmp_req=17 ttl=60 time=4012 ms
64 bytes from 10.0.6.10: icmp_req=18 ttl=60 time=4008 ms
64 bytes from 10.0.6.10: icmp_req=19 ttl=60 time=4013 ms
64 bytes from 10.0.6.10: icmp_req=20 ttl=60 time=4016 ms
```

Figure 24: Ping entre HOME-PC e SERVER com atraso de dois segundos

2.2.2 File Transfer

Serão agora apresentados os testes realizados com *file transfer*. Primeiro foi feito uma transferência de um ficheiro de 10MB do *HOME-PC* para o *SERVER* sem o aumento de atraso:

```
root@HOME-PC:/tmp/pycore.40077/HOME-PC.conf# sudo scp file2.txt core@10.0.6.10:/tm
p/pycore.40077/SERVER.conf/
core@10.0.6.10's password:
file2.txt                                     100%   10MB  74.2KB/s   02:18
```

Figure 25: File transfer entre HOME-PC e SERVER

Incrementamos de seguida o valor do atraso para dois segundos e repetimos a transferência:

```
root@HOME-PC:/tmp/pycore.40077/HOME-PC.conf# sudo scp file2.txt core@10.0.6.10:/
tmp/pycore.40077/SERVER.conf/
core@10.0.6.10's password:
file2.txt                                     100%   10MB  44.7KB/s   03:49
```

Figure 26: File transfer entre HOME-PC e SERVER com atraso de dois segundos

A introdução do atraso teve o efeito antecipado nas ligações. No caso dos *pings* fez com o *round trip time* aumentasse para um valor superior a quatro segundos, visto que a ligação para cada lado terá um atraso mínimo de dois segundos na ligação residencial. Vemos também que o tempo é um bocado superior a quatro segundos devido ao tempo de atraso introduzido pelos outros *links* e por causa do pequeno tempo de transferência do pacote. Relativamente aos *file transfers*, o aumento relativo não foi tão grande mas verificou-se na mesma um atraso significativo, tendo o tempo de transferência total quase duplicado com a introdução do atraso maior na ligação residencial.

3

Após estes testes conseguimos chegar a várias conclusões.

- **Bandwidth**- Não afecta a velocidade de transferência quando os pacotes a transferir são de baixo tamanho comparados com a bandwidth, porém depende da bandwidth mínima da rota usada, ou seja, se a rota usada for composta por diversas ligações a 1Gbps e apenas uma a 512kbps a velocidade de transferência vai depender da ligação com bandwidth mais baixa.
- **Packet Duplication**- Devido á forma como o protocolo TCP opera não causa grandes consequências na velocidade de transferência.
- **Packet Loss**- Quando há perda de pacotes é necessário o reenvio do pacote perdido que aumenta o tempo necessário para transferir todos os pacotes.
- **Delay**- Afecta adversamente as transferências de todos os pacotes e os respectivos ACKS

Tendo em conta estes resultados podemos afirmar que o **Delay** é o factor que mais afecta o tempo necessário para efectuar uma transferência, visto que ao contrário dos outros factores, afecta todos os pacotes enviados independentemente do tamanho.