

Universidade do Minho

Mestrado em Engenharia Informática

Redes Fixas e Móveis

Trabalho Prático

TP1

Grupo 5



Guilherme Soares (A84527)



Maria João Moreira (A89540)



Tiago Rodrigues (A84276)

11 de março de 2022

Conteúdo

1	Estabelecimento das Redes de Acesso e Núcleo	2
1.1	Questão 1	2
1.2	Questão 1.1	3
1.3	Questão 1.2	4
2	Análise de desempenho	6
2.1	Questão 2	6
2.2	Questão 3	7
2.3	Questão 3.1	7
2.4	Questão 3.2	9
2.5	Questão 3.3	10
3	Conclusões	11

1 Estabelecimento das Redes de Acesso e Núcleo

1.1 Questão 1

A **Figura 1** apresenta um exemplo de uma topologia de rede mostrando uma topologia de Rede de Núcleo (CORE Network), onde todas as ligações têm uma capacidade full-duplex (BW) de 1 Gbps e atrasos de 100 μ sec. Existe um servidor **SERVER** (residente no endereço IP 10.0.6.10) directamente ligado à rede de núcleo através de uma ligação Ethernet a 100 Mbps e atraso de 50 μ sec. A rede residencial de acesso usa um Link de Acesso (Access Link - Residential) a 512 Kbps com atraso de 50 ms e o **HOME-PC** tem uma ligação Ethernet ao HOME ROUTER a 100 Mbps e com atraso de 10 μ sec.

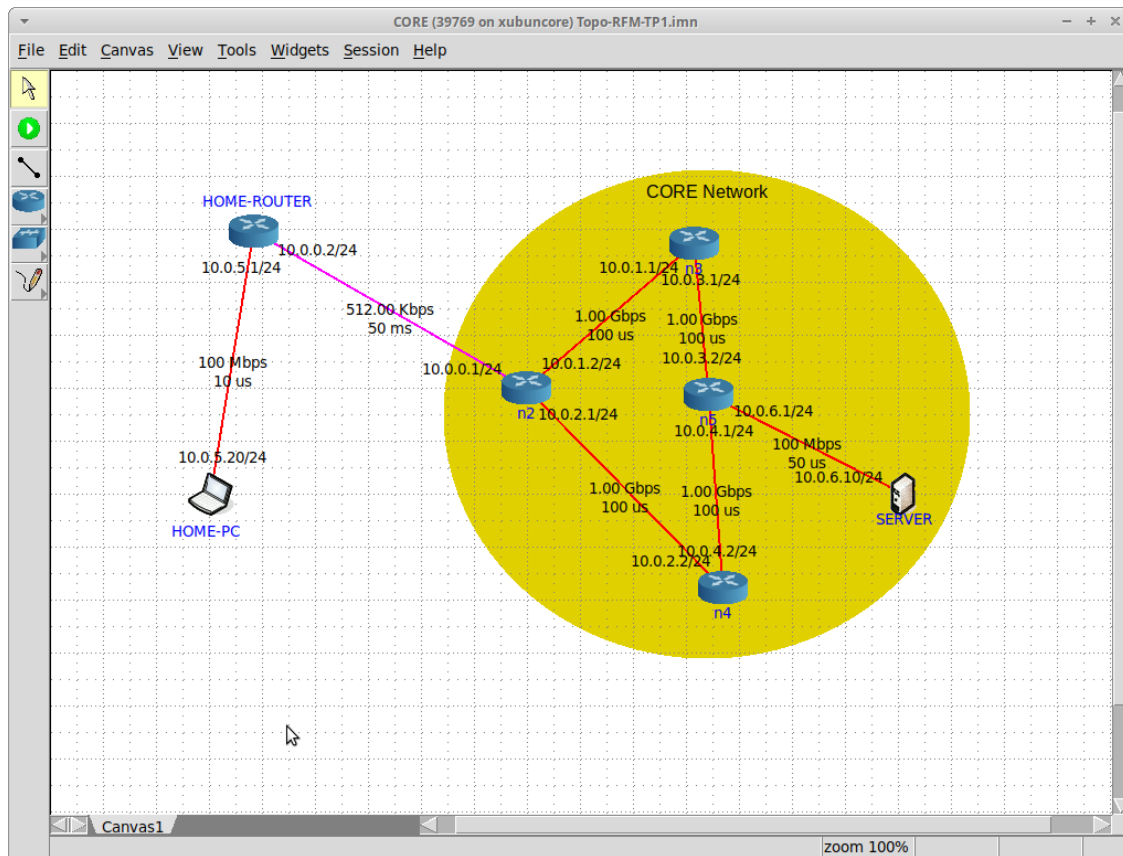


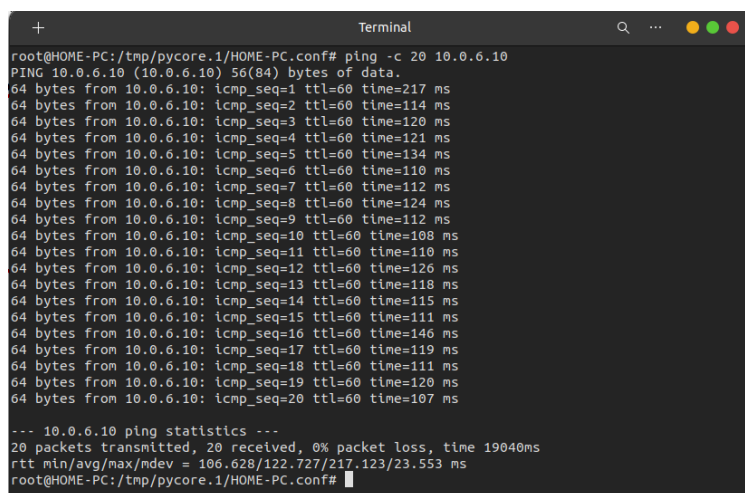
Figura 1: Topologia de rede fornecida no enunciado.

1.2 Questão 1.1

Teste a conectividade global e registre evidências de que está totalmente funcional, utilizando as ferramentas **ping** e **traceroute**. Apresente e justifique o valor de atraso médio – calculado com base nos resultados obtidos com o utilitário ping – entre o **HOME-PC** e o **SERVER**. Verifique, e justifique, se os valores obtidos experimentalmente estão de acordo com os valores teóricos esperados (confrontando-os com as características das redes de Acesso e de Núcleo)

Através das imagens abaixo verificamos que a topologia se encontra totalmente funcional e percebe-se conectividade global, uma vez que o ping realizado conseguiu efetuar o envio de 20 pacotes desde o HOME-PC até ao SERVER.

O envio destes 20 pacotes demorou um total de 19040 ms. A largura de banda e o delay associado entre cada nó faz variar o tempo de atraso, porém, em média, o tempo de atraso registado pelo utilitário **ping** foi de 122.727 ms. Este atraso deve-se ao tamanho dos pacotes enviados, de 64 bytes, e aos valores de atraso definidos entre as ligações de rede. Os valores de atraso registados encontram-se dentro dos valores que esperamos, depois de efetuarmos os cálculos de acordo com a Bandwidth e com valores dos atrasos entre as ligações.



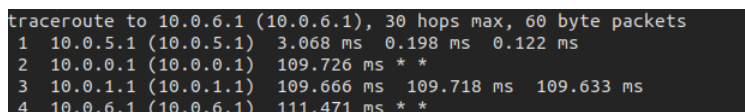
```

+ Terminal
root@HOME-PC:/tmp/pycore.1/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_seq=1 ttl=60 time=217 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=60 time=114 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=60 time=120 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=60 time=121 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=60 time=134 ms
64 bytes from 10.0.6.10: icmp_seq=6 ttl=60 time=110 ms
64 bytes from 10.0.6.10: icmp_seq=7 ttl=60 time=112 ms
64 bytes from 10.0.6.10: icmp_seq=8 ttl=60 time=124 ms
64 bytes from 10.0.6.10: icmp_seq=9 ttl=60 time=112 ms
64 bytes from 10.0.6.10: icmp_seq=10 ttl=60 time=108 ms
64 bytes from 10.0.6.10: icmp_seq=11 ttl=60 time=110 ms
64 bytes from 10.0.6.10: icmp_seq=12 ttl=60 time=126 ms
64 bytes from 10.0.6.10: icmp_seq=13 ttl=60 time=118 ms
64 bytes from 10.0.6.10: icmp_seq=14 ttl=60 time=115 ms
64 bytes from 10.0.6.10: icmp_seq=15 ttl=60 time=111 ms
64 bytes from 10.0.6.10: icmp_seq=16 ttl=60 time=146 ms
64 bytes from 10.0.6.10: icmp_seq=17 ttl=60 time=119 ms
64 bytes from 10.0.6.10: icmp_seq=18 ttl=60 time=111 ms
64 bytes from 10.0.6.10: icmp_seq=19 ttl=60 time=120 ms
64 bytes from 10.0.6.10: icmp_seq=20 ttl=60 time=107 ms

--- 10.0.6.10 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19040ms
rtt min/avg/max/mdev = 106.628/122.727/217.123/23.553 ms
root@HOME-PC:/tmp/pycore.1/HOME-PC.conf#

```

Figura 2: Ping efetuado entre o HOME-PC e o SERVER



```

traceroute to 10.0.6.1 (10.0.6.1), 30 hops max, 60 byte packets
 1  10.0.5.1 (10.0.5.1)  3.068 ms  0.198 ms  0.122 ms
 2  10.0.0.1 (10.0.0.1)  109.726 ms  *  *
 3  10.0.1.1 (10.0.1.1)  109.666 ms  109.718 ms  109.633 ms
 4  10.0.6.1 (10.0.6.1)  111.471 ms  *  *

```

Figura 3: Traceroute efetuado entre o HOME-PC e o SERVER

1.3 Questão 1.2

Agora modifique as características da Ligação de Acesso - Residencial, para que a **taxa de transmissão (BW)** seja actualizada para **1 Gbps / 256 Kbps (1Gbps Downstream; 256 Mbps Upstream)** usando a opção "Configure" nessa ligação (modificações que podem ser feitas sem interromper a execução). Registe, mais uma vez, o valor de atraso médio entre **HOME-PC** e **SERVER** [se estiver a utilizar uma versão do emulador Core abaixo de 4.6 poderá não conseguir testar taxas (BW) assimétricos]. A configuração de ligações assimétricas permite testar ligações residenciais típicas ADSL.

A modificação das características da Ligação de Acesso - Residencial traduziu-se numa diminuição do valor médio de atrasos. Tal acontece, uma vez que ao aumentar a largura de banda é possível a transmissão de mais bytes por segundo, o que torna o envio dos pacotes mais rápido, independentemente de haver os atrasos esperados.

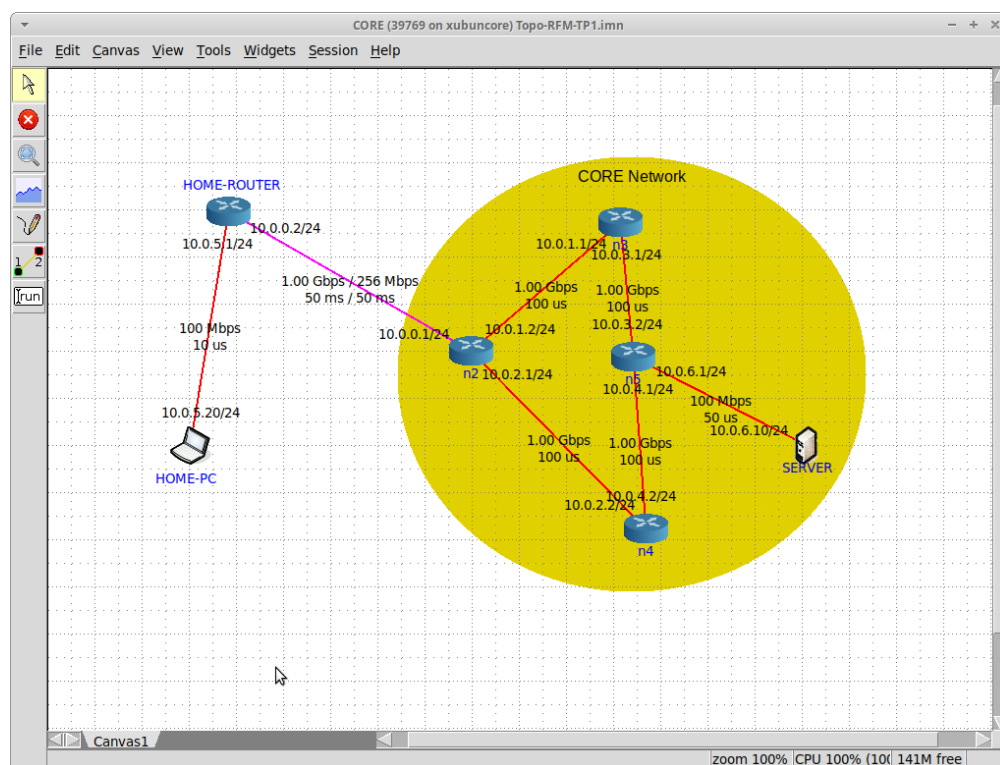
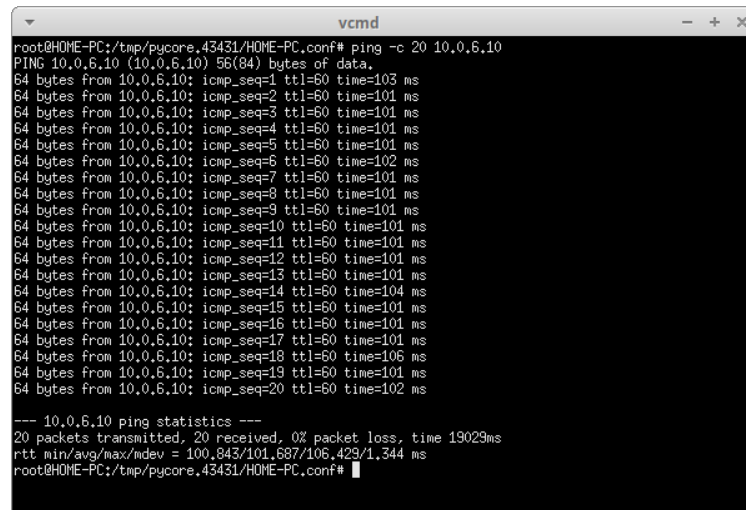


Figura 4: Topologia de rede após as modificações das características da Ligação de Acesso - Residencial.



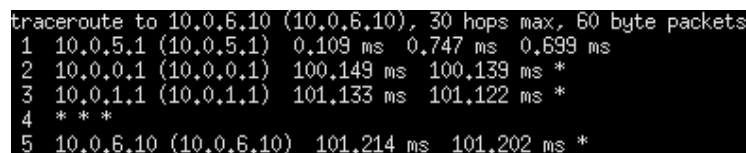
```

vcmd
root@HOME-PC:/tmp/pycore.43431/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_seq=1 ttl=60 time=103 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=6 ttl=60 time=102 ms
64 bytes from 10.0.6.10: icmp_seq=7 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=8 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=9 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=10 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=11 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=12 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=13 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=14 ttl=60 time=104 ms
64 bytes from 10.0.6.10: icmp_seq=15 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=16 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=17 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=18 ttl=60 time=106 ms
64 bytes from 10.0.6.10: icmp_seq=19 ttl=60 time=101 ms
64 bytes from 10.0.6.10: icmp_seq=20 ttl=60 time=102 ms

--- 10.0.6.10 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19029ms
rtt min/avg/max/mdev = 100.843/101.687/106.423/1.344 ms
root@HOME-PC:/tmp/pycore.43431/HOME-PC.conf#

```

Figura 5: Ping efetuado entre o HOME-PC e o SERVER



```

traceroute to 10.0.6.10 (10.0.6.10), 30 hops max, 60 byte packets
 1  10.0.5.1 (10.0.5.1)  0.109 ms  0.747 ms  0.699 ms
 2  10.0.0.1 (10.0.0.1)  100.149 ms  100.139 ms  *
 3  10.0.1.1 (10.0.1.1)  101.133 ms  101.122 ms  *
 4  * * *
 5  10.0.6.10 (10.0.6.10)  101.214 ms  101.202 ms  *

```

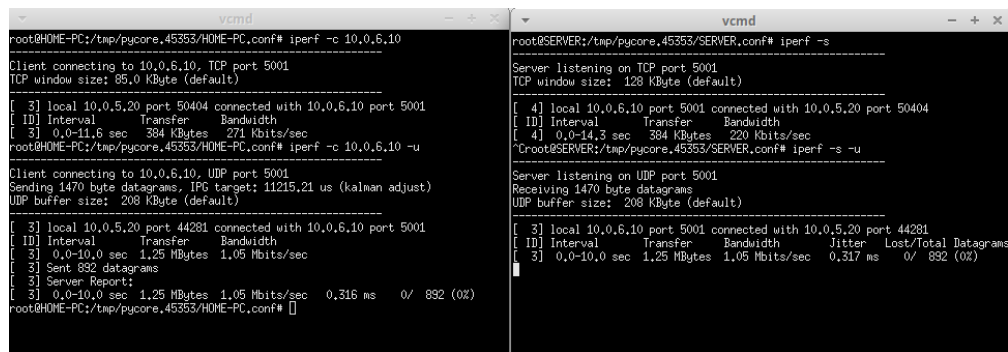
Figura 6: Traceroute efetuado entre o HOME-PC e o SERVER

2 Análise de desempenho

2.1 Questão 2

Analise as características e funcionalidades da ferramenta *iPerf* / *iPerf3* e instale essa ferramenta na plataforma Core. Utilizando “*iperf*” proceda à medição da largura de banda máxima (*BWmax*) atingível em transferências de dados sobre *IP(v4)*, de volume considerável, entre *HOME-PC* e *SERVER*, tanto em *TCP* como em *UDP*.

Utilizando os valores previamente usados na pergunta 1.2 para a Bandwidth no *Access Link - Residencial* (1Gbps de Downstream e 256 Mbps de Upstream), e com o auxílio da ferramenta *iperf* para efetuar a medição da largura de banda máxima atingível na transferência de dados sobre *IP(v4)*, os valores máximos registrados foram os que se apresentam na seguinte figura, tanto para *TCP*, como para *UDP*.



```

vcmd
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 85,0 KByte (default)
-----
[ 3] local 10.0.5.20 port 50404 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-11.6 sec   384 KBytes   221 Kbits/sec
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215,21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.5.20 port 44281 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-10.0 sec   1.25 MBytes   1.05 Mbits/sec
[ 3] Sent 892 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec   1.25 MBytes   1.05 Mbits/sec    0.316 ms    0/ 892 (0%)
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf#

vcmd
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50404
[ ID] Interval      Transfer     Bandwidth
[ 4] 0.0-14.3 sec   384 KBytes   220 Kbits/sec
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 44281
[ ID] Interval      Transfer     Bandwidth    Jitter    Lost/Total Datagrams
[ 3] 0.0-10.0 sec   1.25 MBytes   1.05 Mbits/sec  0.317 ms    0/ 892 (0%)

```

Figura 7: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP.

2.2 Questão 3

Volte ao menu "Configure" do Link de Acesso – Residencial e note que existem vários parâmetros que podem ser configurados com esta ligação, parâmetros como:

- Delay (expresso em microseconds - μsec).
- Loss (expressa pela percentagem de perda de pacotes (controlo aleatório, random).
- Duplicate (expressa pela percentagem de pacotes que, aleatoriamente, são duplicados.)

2.3 Questão 3.1

Modificar um único parâmetro (delay, loss, duplicate) de cada vez, com valores de Loss (valores de 4% e 10%), Duplicate (valores de 2% e 5%) e testar novamente utilizando "iperf" as transferências de dados em UDP e TCP.

The figure shows two terminal windows side-by-side. The left window is titled 'vcmd' and shows the output of 'iperf -c 10.0.6.10' on the HOME-PC. It shows a TCP connection to port 5001 with a bandwidth of 239 Kbits/sec and a UDP connection to port 5001 with a bandwidth of 1.05 Mbits/sec. The right window is also titled 'vcmd' and shows the output of 'iperf -s' on the SERVER. It shows a TCP connection to port 5001 with a bandwidth of 200 Kbits/sec and a UDP connection to port 5001 with a bandwidth of 1.01 Mbits/sec. Both windows show the same results for the 4% loss and 2% duplicate configuration.

```

root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 85.0 KByte (default)
[ 3] local 10.0.5.20 port 50406 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-11.2 sec   325 KBytes  239 Kbits/sec
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.5.20 port 32888 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec   1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 892 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec   1.21 MBytes  1.01 Mbits/sec   0.286 ms  29/ 892 (3.3%)
[ 3] 0.0000-10.0033 sec 14 datagrams received out-of-order
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf#

root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50406
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-13.3 sec   325 KBytes  200 Kbits/sec
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 32888
[ ID] Interval      Transfer    Bandwidth    Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec   1.21 MBytes  1.01 Mbits/sec  0.286 ms  29/ 892 (3.3%)
[ 3] 0.0000-10.0033 sec 14 datagrams received out-of-order

```

Figura 8: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP com os valores de 4% loss e de 2% de pacotes duplicados.

The figure shows two terminal windows side-by-side. The left window is titled 'vcmd' and shows the output of 'iperf -c 10.0.6.10' on the HOME-PC. It shows a TCP connection to port 5001 with a bandwidth of 250 Kbits/sec and a UDP connection to port 5001 with a bandwidth of 1.08 Mbits/sec. The right window is also titled 'vcmd' and shows the output of 'iperf -s' on the SERVER. It shows a TCP connection to port 5001 with a bandwidth of 225 Kbits/sec and a UDP connection to port 5001 with a bandwidth of 1.08 Mbits/sec. Both windows show the same results for the 4% loss and 5% duplicate configuration.

```

root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 85.0 KByte (default)
[ 3] local 10.0.5.20 port 50408 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-11.2 sec   356 KBytes  250 Kbits/sec
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.5.20 port 43191 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec   1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 892 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec   1.28 MBytes  1.08 Mbits/sec   0.267 ms  0/ 892 (0%)
[ 3] 0.0000-10.0033 sec 48 datagrams received out-of-order
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf#

root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50408
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-13.0 sec   356 KBytes  225 Kbits/sec
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 43191
[ ID] Interval      Transfer    Bandwidth    Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec   1.28 MBytes  1.08 Mbits/sec  0.267 ms  0/ 892 (0%)
[ 3] 0.0000-10.0033 sec 48 datagrams received out-of-order

```

Figura 9: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP com os valores de 4% loss e de 5% de pacotes duplicados.


```

vcmd                                     vcmd
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 85.0 KByte (default)
[ 3] local 10.0.5.20 port 50410 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-14.0 sec  234 KBytes  172 Kbits/sec
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.5.20 port 46955 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 892 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec  1.15 MBytes  965 Kbits/sec  0.479 ms  71/ 892 (8%)
[ 3] 0.0000-10.0041 sec  15 datagrams received out-of-order
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf#

root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50410
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-17.0 sec  234 KBytes  142 Kbits/sec
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 46955
[ ID] Interval      Transfer    Bandwidth      Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec  1.15 MBytes  965 Kbits/sec  0.480 ms  71/ 892 (8%)
[ 3] 0.0000-10.0041 sec  15 datagrams received out-of-order

```

Figura 10: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP com os valores de 10% loss e de 2% de pacotes duplicados.

```

vcmd                                     vcmd
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 85.0 KByte (default)
[ 3] local 10.0.5.20 port 50412 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-12.8 sec  325 KBytes  209 Kbits/sec
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.5.20 port 32952 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 892 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec  1.18 MBytes  991 Kbits/sec  0.345 ms  50/ 892 (5.6%)
[ 3] 0.0000-9.9928 sec  31 datagrams received out-of-order
root@HOME-PC:/tmp/pycore.45353/HOME-PC.conf#

root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50412
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-15.6 sec  325 KBytes  171 Kbits/sec
root@SERVER:/tmp/pycore.45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 32952
[ ID] Interval      Transfer    Bandwidth      Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec  1.18 MBytes  991 Kbits/sec  0.346 ms  50/ 892 (5.6%)
[ 3] 0.0000-9.9928 sec  31 datagrams received out-of-order

```

Figura 11: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP com os valores de 10% loss e de 5% de pacotes duplicados.

2.4 Questão 3.2

Para cada combinação de valores (Loss, Duplicate) escolhidos - (4,2), (2,5), (10,2), (10,5)] - determinar, comentar e justificar os resultados obtidos tanto em UDP como TCP.

(4,2) - Neste primeiro cenário, no que toca ao TCP, observamos uma diminuição no intervalo de tempo requerido para a transferência, assim como na quantidade a transferir e na Bandwidth utilizada, uma vez que alguns pacotes foram perdidos, enquanto que outros estão duplicados, sendo assim também mais rápida a sua recuperação. Quanto ao UDP o intervalo de tempo é o mesmo, uma vez que os outros parâmetros também se mantêm quase inalterados, aliado a diminuição do jitter, permite manter o tempo constante.

(4,5) - Aumentando o número de duplicados, no que toca ao TCP verificamos um ligeiro aumento na quantidade a transferir, assim como na Bandwidth utilizada, mantendo-se então o tempo para a transferência praticamente inalterado, apenas com um pequeno incremento do lado do servidor. Em UDP os valores permanecem praticamente constantes, não afetando assim o tempo de transferência.

(10,2) - Com o aumento para 10% de loss, e com o valor de duplicate a 2%, o valor do intervalo de tempo necessário para a transferência excede agora os valores iniciais, uma vez que o Bandwidth é também o mais baixo que verificamos, devido à grande perda de pacotes que decorre. No que toca ao UDP, vemos agora uma maior alteração nos valores obtidos, mas nada significativo, mantendo-se o tempo de transferência inalterado..

(10,4) - Neste último cenário, foi aumentado o número de duplicados em relação ao anterior, resultando assim numa maior Bandwidth, e consequentemente numa maior velocidade na transferência, uma vez que mesmo perdendo muitos pacotes, devido ao facto de haver também um número considerável de duplicados, a perda desses é por vezes facilmente atenuada. Quanto ao UDP, e comparando com o último cenário, temos também uma ligeira melhoria, mas, assim como em todos os cenários de teste, não suficiente para alterar o intervalo de tempo necessário para a transferência.

2.5 Questão 3.3

Modifique finalmente o valor do atraso de ligação para um **valor muito grande** (por exemplo em 2.000.000 μ sec, ou seja 2 segundos (que é um valor próximo do atraso médio Terra-Lua; note p.ex. que o atraso médio Terra-Marte já será de cerca de 10 minutos). Mais uma vez, utilizar “iperf” para testar as características da conectividade fim a fim ”**HOME-PC - SERVER**” em UDP e TCP.

The image shows two terminal windows side-by-side, both titled 'vcmd'. The left window is the HOME-PC terminal, and the right window is the SERVER terminal.

Left Window (HOME-PC):

```
root@HOME-PC:/tmp/pycore,45353/HOME-PC.conf# iperf -c 10.0.6.10
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 65,0 KByte (default)
[ 3] local 10.0.5.20 port 50414 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-10.1 sec  107 KBytes  87.0 Kbits/sec
root@HOME-PC:/tmp/pycore,45353/HOME-PC.conf# iperf -c 10.0.6.10 -u
Client connecting to 10.0.6.10, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215,21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.5.20 port 57977 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 892 datagrams
root@HOME-PC:/tmp/pycore,45353/HOME-PC.conf#
```

Right Window (SERVER):

```
root@SERVER:/tmp/pycore,45353/SERVER.conf# iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 50414
[ ID] Interval      Transfer     Bandwidth
[ 4] 0.0-10.3 sec  107 KBytes  10.2 Kbits/sec
root@SERVER:/tmp/pycore,45353/SERVER.conf# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
[ 3] local 10.0.6.10 port 5001 connected with 10.0.5.20 port 57977
[ ID] Interval      Transfer     Bandwidth      Jitter   Lost/Total Datagrams
[ 3] 0.0-10.0 sec  1.17 MBytes  983 Kbits/sec  0.639 ms  56/ 892 (6.3%)
root@SERVER:/tmp/pycore,45353/SERVER.conf#
```

Figura 12: IPerf efetuado entre o HOME-PC e o SERVER em TCP e UDP com o valor do atraso de ligação muito grande.

Como podemos observar através das imagens, este enorme aumento no atraso de ligação vai afetar gravemente o TCP, como tínhamos vindo a observar na questão anterior, enquanto que o UDP novamente não é alterado.

3 Conclusões

O presente trabalho prático tem como objetivo estudar características de desempenho das redes com serviço BE (Best Effort) das comunicações na Internet, utilizando para esta finalidade o emulador de rede Core e a ferramenta iPerf.

Existem vários fatores que afetam o desempenho de redes, dentre eles estão a largura de banda, perda de pacotes e duplicação de pacotes, que foram o alvo de estudo deste trabalho prático.

Em primeiro lugar, observamos que o valor dos atrasos influencia diretamente na quantidade de tempo que um pacote demora a ser enviado desde a sua fonte até ao seu destino, logo a variação deste valor pode causar o aumento ou diminuição do tempo de entrega de um pacote de dados. No entanto no exercício 3.3 notamos que atribuir um valor de delay muito grande tem um efeito notável no TCP ao contrário do UDP que parece não sofrer com este aumento.

Outro parâmetro que está relacionado de forma direta com o tempo de atraso de entrega é a largura de banda, esta limita a quantidade de dados que pode ser transmitida num segundo. Como observamos na questão 1, aumentar o valor de largura de banda diminuiu os atrasos, ou seja deu-se um aumento da capacidade de transmissão, num segundo mais dados conseguem ser enviados, então a transmissão do pacote consegue ser efetuada de forma mais rápida diminuindo assim o valor dos atrasos.

Percebemos que as perdas de pacotes podem atrasar mais a entrega de pacotes, bem como aumentar o número de pacotes duplicados pode também influenciar negativamente o desempenho da entrega.

Verificamos através do exercício 3 que o valor de pacotes que se perdem e o valor de pacotes duplicados possuem uma relação, quando acontecem perdas de pacotes, a duplicação de pacotes pode suprir esta perda e atenuar o atraso causado pelas perdas.

Por fim, percebemos através dos testes realizados que a alteração dos parâmetros influencia mais o desempenho de transmissão para TCP do que para UDP.