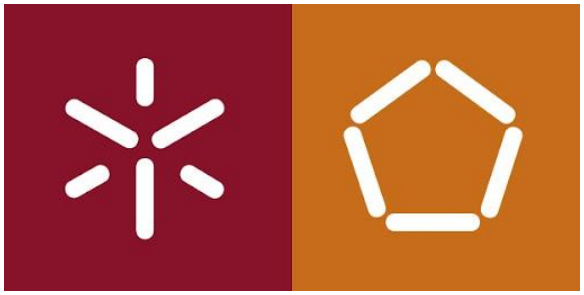


Universidade do Minho

Relatório do Trabalho 1

Ano letivo 2022/2023

Outubro 2022

Mestrado em Engenharia Informática

Unidade Curricular de Engenharia de Sistemas de Rede

Grupo 80

Ana Catarina Gonçalves PG50180

Francisco Toldy PG50379

Pedro Araújo PG50684

Índice

1. Introdução

Este trabalho prático foi desenvolvido no âmbito da unidade curricular de Engenharia de Serviços em Rede que tem como objetivo de investigação e familiarização com serviços de Streaming, dividido em 3 etapas.

A primeira etapa consiste principalmente na gravação de um pequeno vídeo e a sua transmissão. Com esta base no servidor, vão ser criados clientes (VLC, firefox e ffplay) que vão ter acesso ao vídeo transmitido. Assim sendo, vai ser investigado o débito de informação dependendo do número de clientes, e explorar o efeito do aumento deste número de clientes.

Na sequência da etapa um, a etapa 2 consistia na criação de 3 videos com qualidade cada vez menor, a partir do video B. Com o objetivo de obrigar qualquer cliente que não tenha largura de banda o suficiente para obter o video de melhor qualidade, receber um que a sua largura de banda o permitisse.

A terceira etapa passou por primeiramente fazer streaming do video A em unicast para um dos portáteis na topologia e depois criar uma topologia nova e fazer um teste de streaming com multicast para 4 portáteis em simultâneo. Esta etapa teria como objetivo fazer comparação entre as duas de transmissão, nomeadamente a escalabilidade e gasto dados.

Etapa 1

Passo 1

Construa uma topologia base no CORE, com pelo menos um servidor, 3 portáteis, 2 switches e um ou dois routers, mais ou menos como sugerido na figura.

Foi criada a topologia no core, seguindo o exemplo da figura que estava apresentada no enunciado

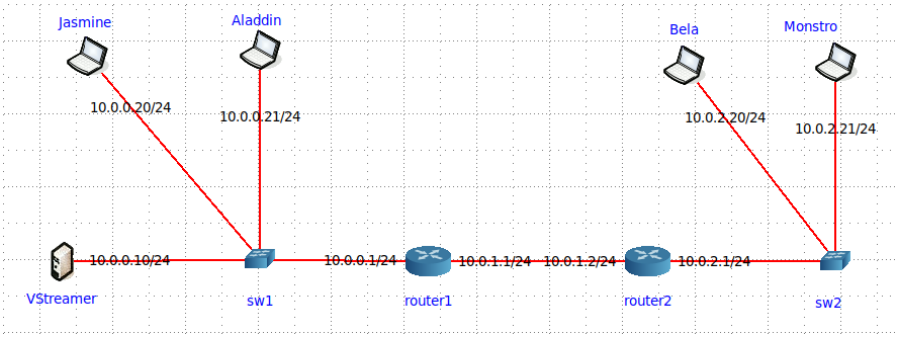


Figura 1 - Topologia Criada

Passo 2

Teste a conectividade da rede, com ping, traceroute ou outros comandos, para ter a certeza que funciona tudo bem

Foram realizados testes de conexão entre alguns pontos da rede utilizando os comandos ping e traceroute:

```
root@Jasmine:/tmp/pycore.38911/Jasmine.conf# ping 10.0.2.21
PING 10.0.2.21 (10.0.2.21) 56(84) bytes of data.
64 bytes from 10.0.2.21: icmp_seq=1 ttl=64 time=0.071 ms
64 bytes from 10.0.2.21: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.0.2.21: icmp_seq=3 ttl=64 time=0.070 ms
^C
--- 10.0.2.21 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.067/0.069/0.071/0.001 ms
root@Jasmine:/tmp/pycore.38911/Jasmine.conf#
```

Figura 2 - Ping de Jasmine para Aladdin

```
root@Jasmine:/tmp/pycore.38911/Jasmine.conf# ping 10.0.4.20
PING 10.0.4.20 (10.0.4.20) 56(84) bytes of data.
64 bytes from 10.0.4.20: icmp_seq=1 ttl=62 time=0.126 ms
64 bytes from 10.0.4.20: icmp_seq=2 ttl=62 time=0.116 ms
64 bytes from 10.0.4.20: icmp_seq=3 ttl=62 time=0.112 ms
^C
--- 10.0.4.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.112/0.118/0.126/0.005 ms
root@Jasmine:/tmp/pycore.38911/Jasmine.conf#
```

Figura 3 - Ping de Jasmine para Bella

```
root@Aladdin:/tmp/pycore.38911/Aladdin.conf# traceroute 10.0.2.20
traceroute to 10.0.2.20 (10.0.2.20), 30 hops max, 60 byte packets
 1  10.0.2.20 (10.0.2.20)  0.041 ms  0.007 ms  0.006 ms
root@Aladdin:/tmp/pycore.38911/Aladdin.conf#
```

Figura 4 - Traceroute de Aladdin para Jasmine

```
root@Bela:/tmp/pycore.42171/Bela.conf# traceroute 10.0.2.21
traceroute to 10.0.2.21 (10.0.2.21), 30 hops max, 60 byte packets
 1  10.0.2.21 (10.0.2.21)  0.065 ms  0.006 ms  0.006 ms
root@Bela:/tmp/pycore.42171/Bela.conf#
```

Figura 5 - Traceroute de Bella para Monstro

Passo 3

Abra uma bash no servidor VStreamer e, seguindo as indicações do ANEXO 2, coloque o VLC a fazer streaming por HTTP do ficheiro videoA.mp4 com transcoding para "Video – Theora + Vorbis (Ogg)". VLC à Media à Stream à ...

Foram seguidos os passos no anexo 2, como se pode verificar o resultado na seguinte imagem:

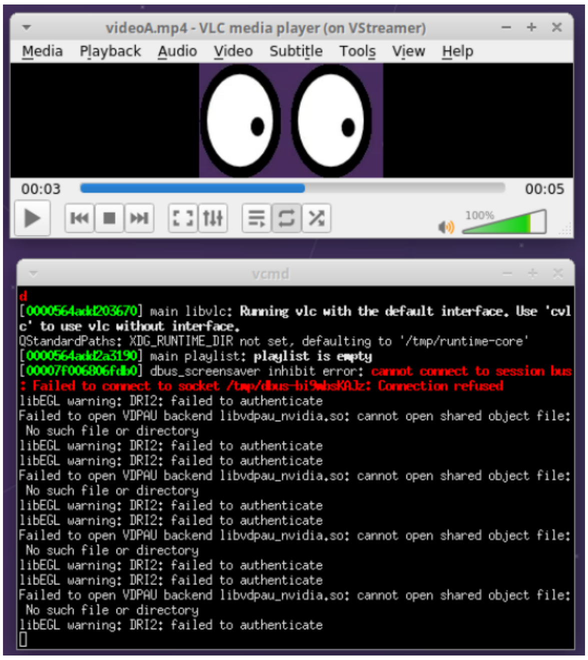


Figura 6 - Servidor a transmitir videoA em loop

Passo 4

Abra uma nova bash no portátil Jasmine, configure o DISPLAY e coloque um segundo VLC a funcionar agora como cliente: Media à Open Network Stream à http://10.0.0.10:8080/

Mais uma vez, foram seguidos os passos sendo o resultado o seguinte:

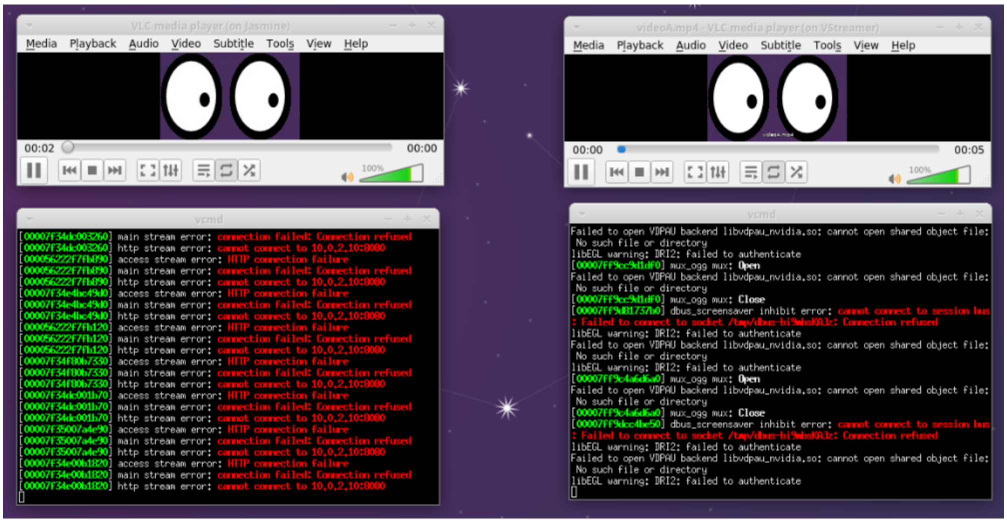


Figura 7- Visualização da Stream na Jasmine à esquerda e partilha do Video no servidor à direita

Passo 5

Recolha uma amostra de tráfego com o Wireshark na interface de saída do servidor.

Foi recolhida a amostra seguinte de trafego com apenas um cliente a assistir a transmissão com vcl:

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=1 Ack=1 Win=509 Len=1448 TSval=4180953…
2	0.000000970	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=1449 Ack=1 Win=509 Len=1448 TSval=4180…
3	0.000001328	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=2897 Ack=1 Win=509 Len=1448 TSval=4180…
4	0.000001658	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=4345 Ack=1 Win=509 Len=1448 TSval=4180…
5	0.000001985	10.0.0.10	10.0.0.20	TCP	364	8080 → 57928 [PSH, ACK] Seq=5793 Ack=1 Win=509 Len=298 TSval=…
6	0.000077287	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=1449 Win=501 Len=0 TSval=3829927…
7	0.000080338	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=2897 Win=496 Len=0 TSval=3829927…
8	0.000082182	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=4345 Win=491 Len=0 TSval=3829927…
9	0.000083978	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=5793 Win=485 Len=0 TSval=3829927…
10	0.000085776	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=6091 Win=484 Len=0 TSval=3829927…
11	0.492151364	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=6091 Ack=1 Win=509 Len=1448 TSval=4180…
12	0.492152234	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=7539 Ack=1 Win=509 Len=1448 TSval=4180…
13	0.492152609	10.0.0.10	10.0.0.20	TCP	1370	8080 → 57928 [PSH, ACK] Seq=8987 Ack=1 Win=509 Len=1304 TSval…
14	0.492210641	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=7539 Win=501 Len=0 TSval=3829928…
15	0.492213775	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=8987 Win=496 Len=0 TSval=3829928…
16	0.492215626	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=10291 Win=491 Len=0 TSval=382992…
18	0.758096798	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=10291 Ack=1 Win=509 Len=1448 TSval=418…
19	0.758098273	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=11739 Ack=1 Win=509 Len=1448 TSval=418…
20	0.758098603	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=13187 Ack=1 Win=509 Len=1448 TSval=418…
21	0.758098911	10.0.0.10	10.0.0.20	TCP	182	8080 → 57928 [PSH, ACK] Seq=14635 Ack=1 Win=509 Len=116 TSval…
22	0.758159356	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=11739 Win=501 Len=0 TSval=382992…
23	0.758162736	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=13187 Win=496 Len=0 TSval=382992…
24	0.758164462	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=14635 Win=491 Len=0 TSval=382992…
25	0.758166147	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=14751 Win=491 Len=0 TSval=382992…
26	1.249761785	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=14751 Ack=1 Win=509 Len=1448 TSval=418…
27	1.249762616	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=16199 Ack=1 Win=509 Len=1448 TSval=418…
28	1.249762948	10.0.0.10	10.0.0.20	TCP	1435	8080 → 57928 [PSH, ACK] Seq=17647 Ack=1 Win=509 Len=1369 TSva…
29	1.249817707	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=16199 Win=501 Len=0 TSval=382992…
30	1.249820580	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=17647 Win=496 Len=0 TSval=382992…
31	1.249822302	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=19016 Win=491 Len=0 TSval=382992…
32	1.736041153	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=19016 Ack=1 Win=509 Len=1448 TSval=418…
33	1.736042158	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=20464 Ack=1 Win=509 Len=1448 TSval=418…
34	1.736042517	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=21912 Ack=1 Win=509 Len=1448 TSval=418…
35	1.736042850	10.0.0.10	10.0.0.20	TCP	471	8080 → 57928 [PSH, ACK] Seq=23360 Ack=1 Win=509 Len=405 TSval…
36	1.736108852	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=20464 Win=501 Len=0 TSval=382992…
37	1.736111876	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=21912 Win=496 Len=0 TSval=382992…
38	1.736113738	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=23360 Win=491 Len=0 TSval=382992…
39	1.736115550	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=23765 Win=489 Len=0 TSval=382992…
40	2.009002108	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=23765 Ack=1 Win=509 Len=1448 TSval=418…
41	2.009003409	10.0.0.10	10.0.0.20	TCP	1514	8080 → 57928 [ACK] Seq=25213 Ack=1 Win=509 Len=1448 TSval=418…
42	2.009003882	10.0.0.10	10.0.0.20	TCP	1451	8080 → 57928 [PSH, ACK] Seq=26661 Ack=1 Win=509 Len=1385 TSva…
43	2.009076157	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=25213 Win=501 Len=0 TSval=382992…
44	2.009079978	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=26661 Win=496 Len=0 TSval=382992…
45	2.009082251	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=28046 Win=491 Len=0 TSval=382992…
47	2.728099108	10.0.0.10	10.0.0.20	TCP	66	8080 → 57928 [FIN, ACK] Seq=28046 Ack=1 Win=509 Len=0 TSval=4…
48	2.770261722	10.0.0.20	10.0.0.10	TCP	66	57928 → 8080 [ACK] Seq=1 Ack=28047 Win=501 Len=0 TSval=382993…

Figura 8 - Amostra de Trafego de Wireshark apenas com uma stream (VLC)

Passo 7

No portátil Bela, abra uma bash, configure o DISPLAY, e execute o Firefox. Abra a página video.html criada e verifique se o vídeo aparece corretamente. Ajuste o HTML se necessário.

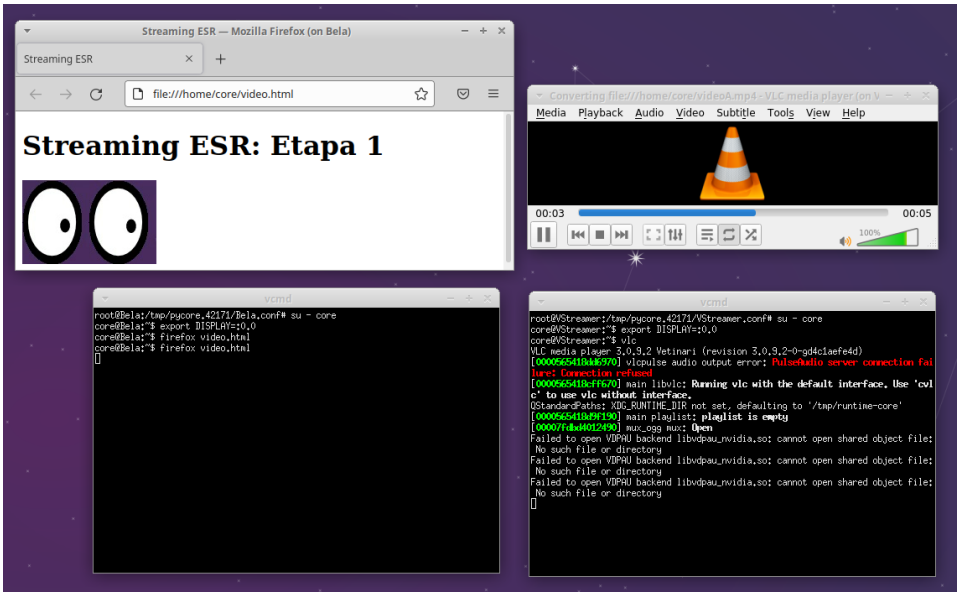


Figura 9 - Visualização do vídeo através do Firefox na Bela (à esquerda) e o servidor (a transmitir à direita)

Passo 9

No portátil Monstro, abra uma bash, configure o DISPLAY e use o comando ffplay http://:8080/ como terceiro cliente da stream de video.

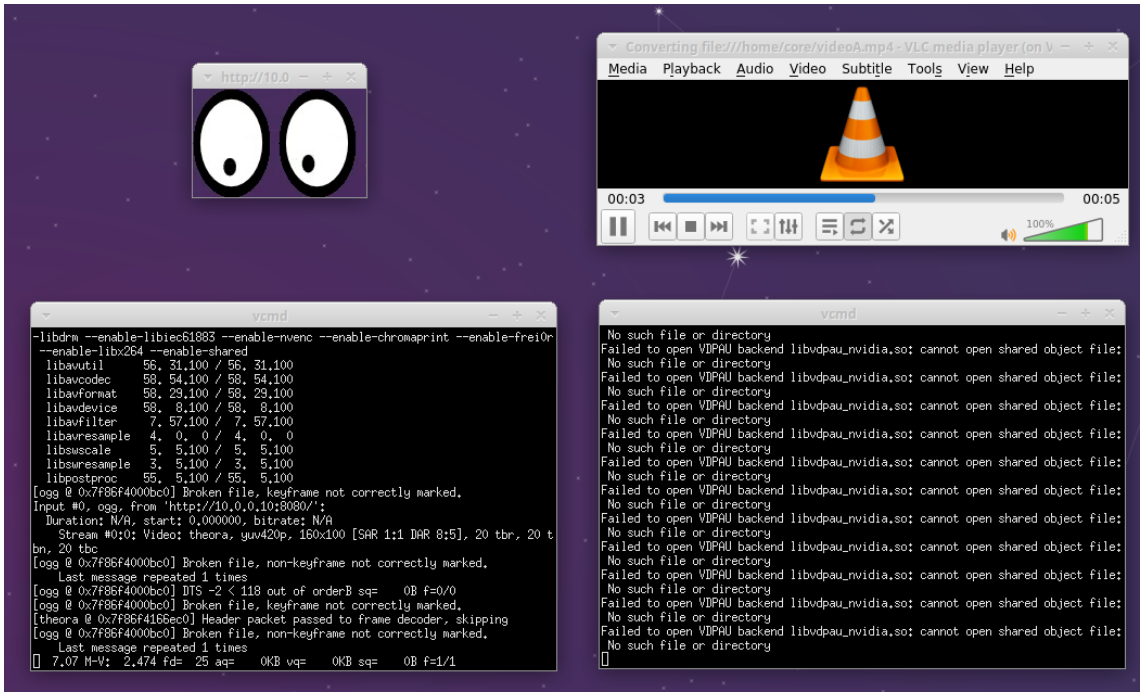


Figura 10 - Visualização do vídeo através do ffplay no Monstro à esquerda e o servidor a transmitir à direita

Questão 1

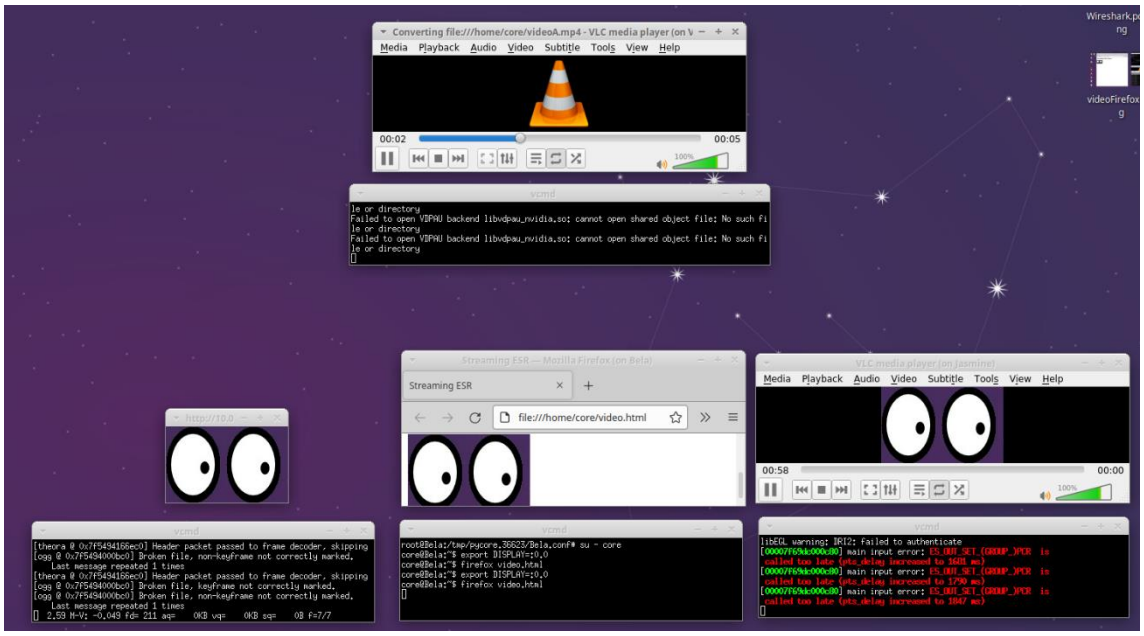


Figura 11- Visualização do vídeo através do FFPLAY no Monstro à esquerda, o servidor em cima, a Bela no Firefox no meio e a Jasmine com o VCL à direita

Analisando os resultados obtidos nos passos anteriores chegamos as seguintes conclusões:

- No que toca a bps:
  - Taxa teórica de bps necessária : 14 kbps

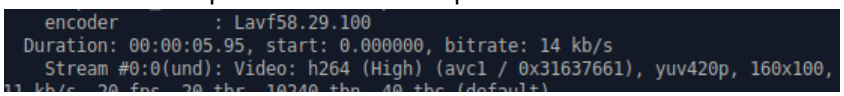


Figura 12 - Captura do comando "ffmpeg -i videoA.mp4"

  - Taxa efetiva a usar apenas a usar o VCL : 122 kbps

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	62	100.0	44137	128 k
Ethernet	100.0	62	2.0	868	2526
Internet Protocol Version 4	100.0	62	2.8	1240	3609
Transmission Control Protocol	100.0	62	95.2	42029	122 k
Hypertext Transfer Protocol	3.2	2	6.6	2896	8431

Figura 13 - Débito ao transmitir para 1 stream: VCL

  - Taxa efetiva de bps apenas a usar o Firefox : 11 kbps

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	67	100.0	44515	84 k
Ethernet	100.0	67	2.1	938	1786
Internet Protocol Version 6	1.5	1	0.1	40	76
Open Shortest Path First	1.5	1	0.1	36	68
Internet Protocol Version 4	98.5	66	3.0	1320	2514
Transmission Control Protocol	95.5	64	94.6	42093	80 k
Hypertext Transfer Protocol	6.0	4	13.0	5792	11 k
Open Shortest Path First	3.0	2	0.2	88	167

Figura 14 - Débito ao transmitir para 1 stream: Firefox

  - Taxa efetiva a usar apenas o FFPLAY: 120kbs

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	65	100.0	44383	127 k
Ethernet	100.0	65	2.1	910	2605
Internet Protocol Version 6	1.5	1	0.1	40	114
Open Shortest Path First	1.5	1	0.1	36	103
Internet Protocol Version 4	98.5	64	2.9	1280	3664
Transmission Control Protocol	95.4	62	94.7	42029	120 k
Hypertext Transfer Protocol	3.1	2	6.5	2896	8290
Open Shortest Path First	3.1	2	0.2	88	251

Figura 15- Débito ao transmitir para 1 stream: FFPLAY

  - Taxa efetiva com Firefox e VLC em simultâneo : 71 kbps

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	E
Frame	100.0	48	100.0	34553	75 k	0
Ethernet	100.0	48	1.9	672	1467	0
Internet Protocol Version 4	100.0	48	2.8	960	2096	0
Transmission Control Protocol	100.0	48	95.3	32921	71 k	4
Hypertext Transfer Protocol	2.1	1	4.2	1448	3161	1

Figura 16 - Débito ao transmitir para 2 streams

  - Taxa com os 3 em simultâneo : 7522 bps

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s
Frame	100.0	38	100.0	27803	144 k
Ethernet	100.0	38	1.9	532	2763
Internet Protocol Version 4	100.0	38	2.7	760	3948
Transmission Control Protocol	100.0	38	95.4	26511	137 k
Hypertext Transfer Protocol	2.6	1	5.2	1448	7522

Figura 17 - Débito ao transmitir para 3 streams

Relativamente ao encapsulamento usado, podemos concluir que foi utilizado o TCP para encapsular os pacotes enviados por HTTP. Isto é comprovado por qualquer das imagens anteriores relativamente aos bps, onde se pode identificar acima do protocolo HTTP o protocolo TCP.

Em termos de fluxos, verificamos através de filtros que existiam 3 fluxos (recorrendo ao filtro ’tcp.stream eq 0/1/2’).

Finalmente, podemos concluir que, o aumento de clientes resulta numa diminuição linear do débito. Logo, a escalabilidade é demasiado reduzida nesta solução, visto que em streams com vários clientes seria praticamente impossível de partilhar o vídeo devido a largura de banda disponível ser demasiado pequena.

Etapa 2

Nesta etapa, foram seguidos todos os passos até ao passo 9 para preparar o ambiente para o teste na topologia, que agrupava a criação dos ficheiros com menor qualidade, a produção do ficheiro MPD e a página html que transmitirá o video com a devida qualidade.

Passo 9

Visualizar com o firefox no portátil Bela.

Figura 18 - Terminal da Bela (cima) . Terminal do servidor a fornecer o serviço (em baixo) e Transmissão na Bela do videoB com maior qualidade

Passo 10

Visualizar com o firefox no portátil Alladin.

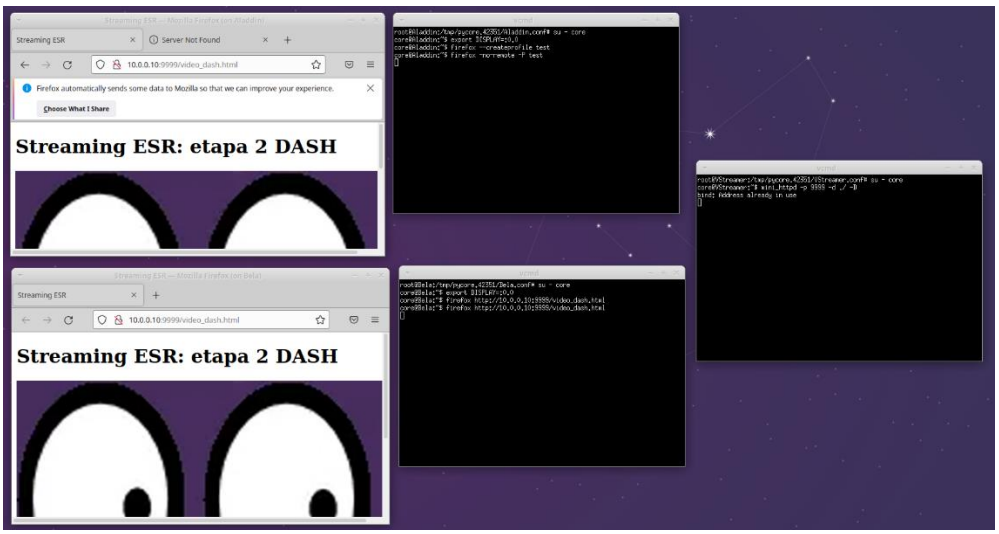


Figura 19 - Perfil do Alladin (cima) , Bela (baixo) e à Servidor (direita)

Passo 12

Mexer na capacidade dos links de modo a forçar o portátil Bela a mostrar o vídeo de menor dimensão

Para efetivamente verificar se a troca de videos dependendo da largura de banda efetivamente acontece, alteramos o valor da largura de banda disponível para o portátil Bela (para 400kps) e verificamos que de facto acontece, e é transmitido o video com menor qualidade (e também menor tamanho)

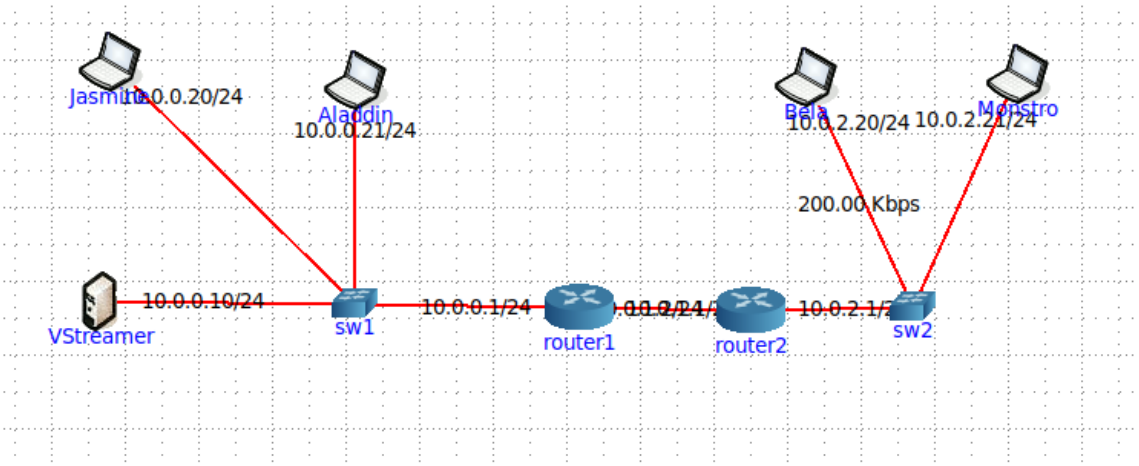


Figura 20 - Alteração realizada na topologia encontra-se no link entre a Bela e o switch 2 para 200kbps

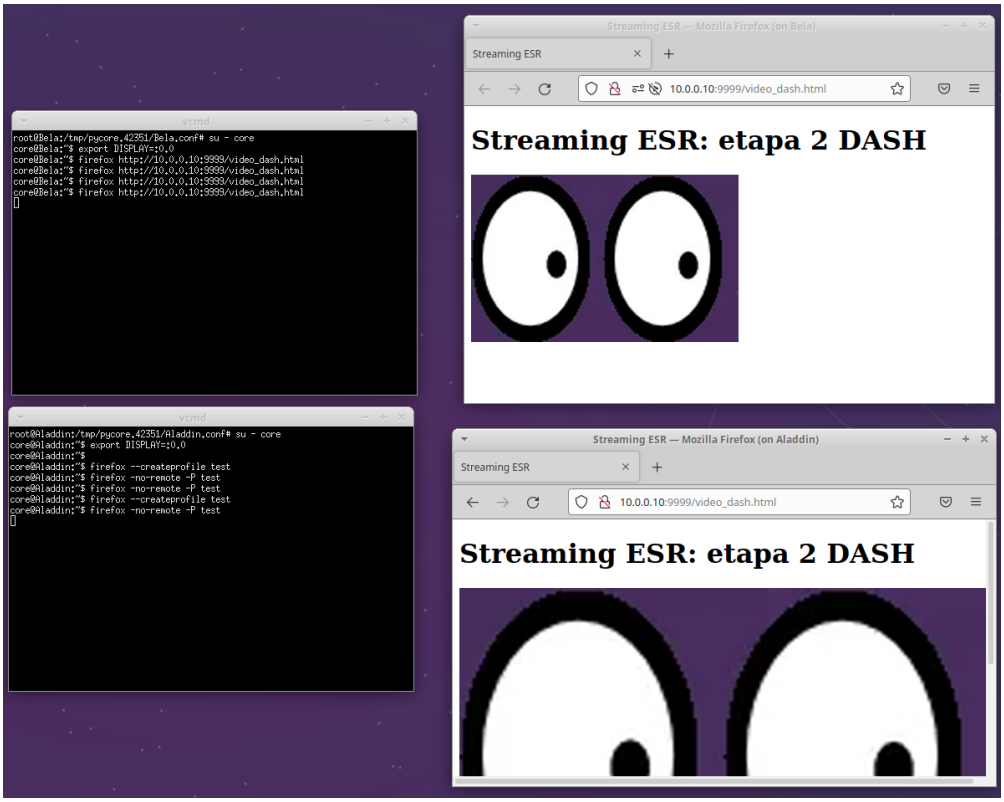


Figura 21 - Com 400bps: Em cima encontra-se a transmissão para o portátil Bela (menor qualidade e tamanho) e em baixo o portátil Aladdin (maior qualidade e tamanho)

Com uma redução maior (200 kbps) o vídeo transmitido é aquele com menor qualidade e tamanho:

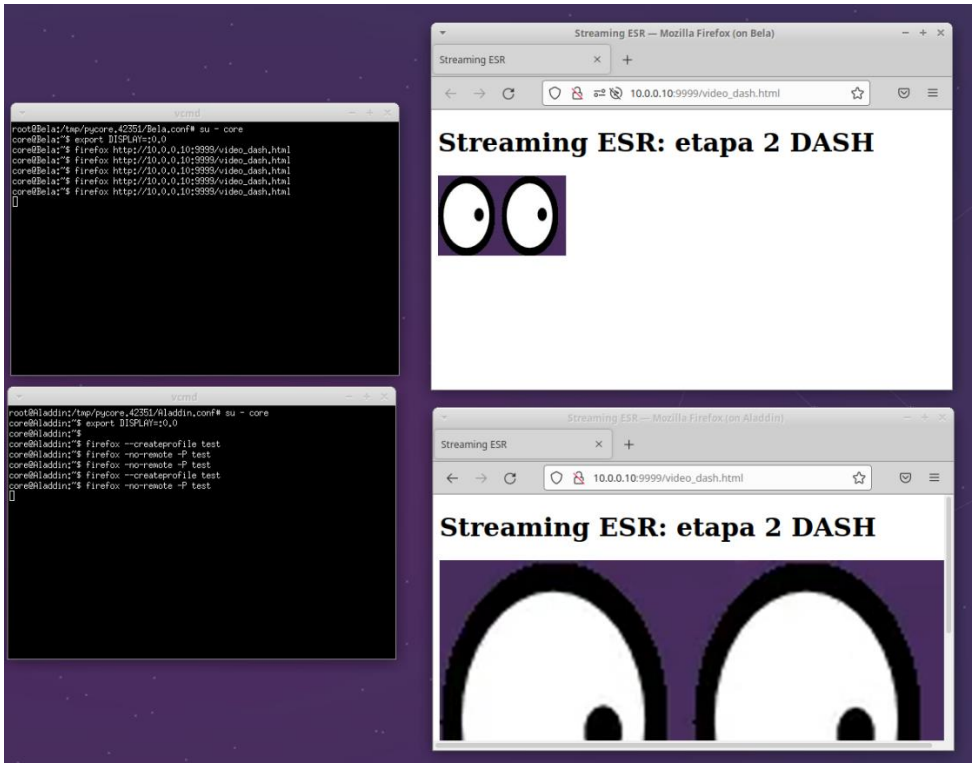
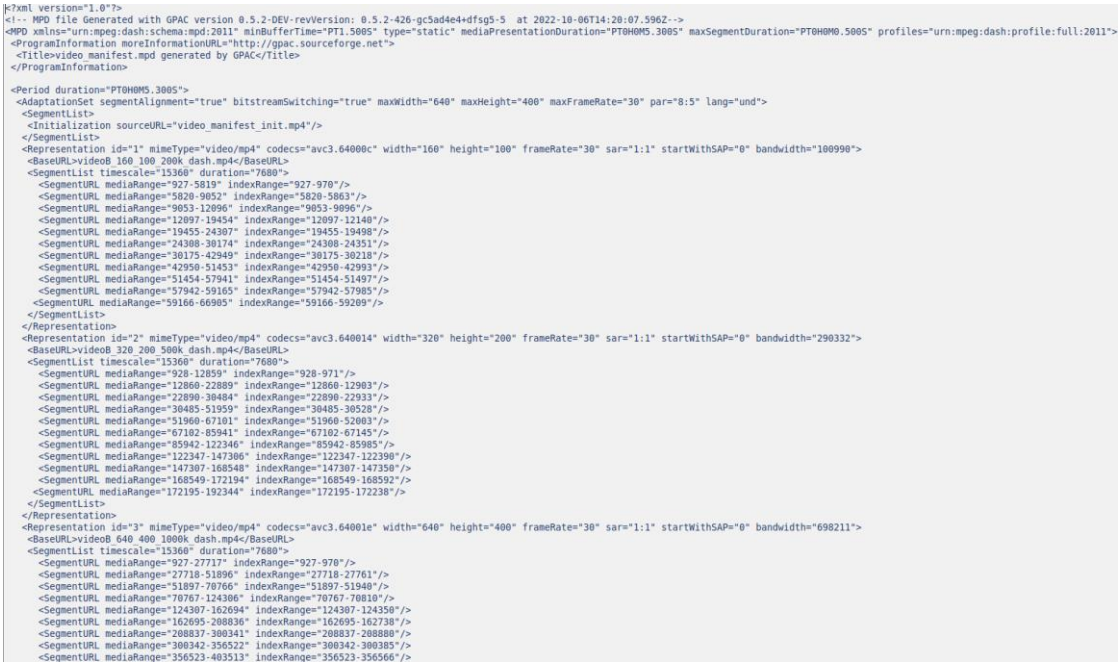


Figura 22 - Com 200kps: Em cima encontra-se a transmissão para o portátil Bela (tamanho e qualidade mais reduzido) e em baixo o portátil Aladdin (maior qualidade e tamanho).

Questão 2

Diga qual a largura de banda necessária, em bits por segundo, para que o cliente de streaming consiga receber o vídeo no firefox e qual a pilha protocolar usada neste cenário

A largura banda mínima necessária , de acordo com o ficheiro manifest é 100900 bits por segundo, que corresponde ao video com menor qualidade e tamanho. Em termos de pilha protocolar, foi utilizado HTTP sobre TCP sobre IP.



```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gc5ad4e4+dfsg5-5  at 2022-10-06T14:20:07.596Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H0M5.300S" maxSegmentDuration="PT0H0M0.500S" profiles="urn:mpeg:dash:profile:full:2011">
 <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
  <Title>video_manifest.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H0M5.300S">
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="640" maxHeight="400" maxFrameRate="30" par="8:5" lang="und">
   <SegmentList>
    <Initialization sourceURL="video_manifest_init.mp4"/>
   </SegmentList>
   <Representation id="1" mimeType="video/mp4" codecs="avc3.64000c" width="160" height="100" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="100990">
    <BaseURL>videoB_160_100_200k_dash.mp4</BaseURL>
    <SegmentList timescale="15360" duration="7680">
      <SegmentURL mediaRange="927-5819" indexRange="927-970"/>
      <SegmentURL mediaRange="5820-9052" indexRange="5820-5863"/>
      <SegmentURL mediaRange="9053-12096" indexRange="9053-9096"/>
      <SegmentURL mediaRange="12097-19454" indexRange="12097-12140"/>
      <SegmentURL mediaRange="19455-24307" indexRange="19455-19498"/>
      <SegmentURL mediaRange="24308-30174" indexRange="24308-24351"/>
      <SegmentURL mediaRange="30175-42949" indexRange="30175-30218"/>
      <SegmentURL mediaRange="42950-51453" indexRange="42950-42993"/>
      <SegmentURL mediaRange="51454-57941" indexRange="51454-51497"/>
      <SegmentURL mediaRange="57942-59165" indexRange="57942-57985"/>
     <SegmentURL mediaRange="59166-66905" indexRange="59166-59209"/>
    </SegmentList>
   </Representation>
   <Representation id="2" mimeType="video/mp4" codecs="avc3.640014" width="320" height="200" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="290332">
    <BaseURL>videoB_320_200_500k_dash.mp4</BaseURL>
    <SegmentList timescale="15360" duration="7680">
      <SegmentURL mediaRange="928-12859" indexRange="928-971"/>
      <SegmentURL mediaRange="12860-22889" indexRange="12860-12903"/>
      <SegmentURL mediaRange="22890-30484" indexRange="22890-22933"/>
      <SegmentURL mediaRange="30485-51959" indexRange="30485-30528"/>
      <SegmentURL mediaRange="51960-67101" indexRange="51960-52003"/>
      <SegmentURL mediaRange="67102-85941" indexRange="67102-67145"/>
      <SegmentURL mediaRange="85942-122346" indexRange="85942-85985"/>
      <SegmentURL mediaRange="122347-147306" indexRange="122347-122390"/>
      <SegmentURL mediaRange="147307-168548" indexRange="147307-147350"/>
      <SegmentURL mediaRange="168549-172194" indexRange="168549-168592"/>
     <SegmentURL mediaRange="172195-192344" indexRange="172195-172238"/>
    </SegmentList>
   </Representation>
   <Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="400" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="698211">
    <BaseURL>videoB_640_400_1000k_dash.mp4</BaseURL>
    <SegmentList timescale="15360" duration="7680">
      <SegmentURL mediaRange="927-27717" indexRange="927-970"/>
      <SegmentURL mediaRange="27718-51896" indexRange="27718-27761"/>
      <SegmentURL mediaRange="51897-70766" indexRange="51897-51940"/>
      <SegmentURL mediaRange="70767-124306" indexRange="70767-70810"/>
      <SegmentURL mediaRange="124307-162694" indexRange="124307-124350"/>
      <SegmentURL mediaRange="162695-208836" indexRange="162695-162738"/>
      <SegmentURL mediaRange="208837-300341" indexRange="208837-208880"/>
      <SegmentURL mediaRange="300342-356522" indexRange="300342-300385"/>
      <SegmentURL mediaRange="356523-403513" indexRange="356523-356566"/>
```


Figura 23 - Ficheiro manifesto gerado

Questão 3

Ajuste o débito dos links da topologia de modo que o cliente no portátil Bela exiba o vídeo de menor resolução e o cliente no portátil Alladin exiba o vídeo com mais resolução. Mostre evidências

Como foi verificado anteriormente (Figura 22), o débito na topologia foi alterado para 200kbps no link entre a Bela e o switch 2, e por isso, foi possível verificar a transmissão do video com menor resolução existente. Além disso, manteve-se a transmissão no portátil do Aladdin para permitir uma comparação entre os 2 videos.

Questão 4

Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.

Neste caso, o ficheiro manifest gerado possui informação dos chunks existentes de cada versão de encoding do video e da largura de banda mínima necessária para a transmissão de cada um. Assim, o cliente , quando pedir os chunks do video, obtêm informação sobre quais pedir de acordo com a largura de banda que esteve disponível no request do chunk anterior, por default vai ser requisitado o chunk com maior qualidade. Como pudemos observar, quando diminuímos a largura de banda da ligação do Bela para 200 kbps (entre o mínimo da qualidade mais baixa e o mínimo da qualidade média) pudemos observar que a reprodução foi a mais baixa qualidade.

Etapa 3

Passo 2 e 3

No servidor VStreamer, inicie uma sessão de streaming com RTP com o ffmpeg. No cliente Monstro, inicie um cliente ffplay

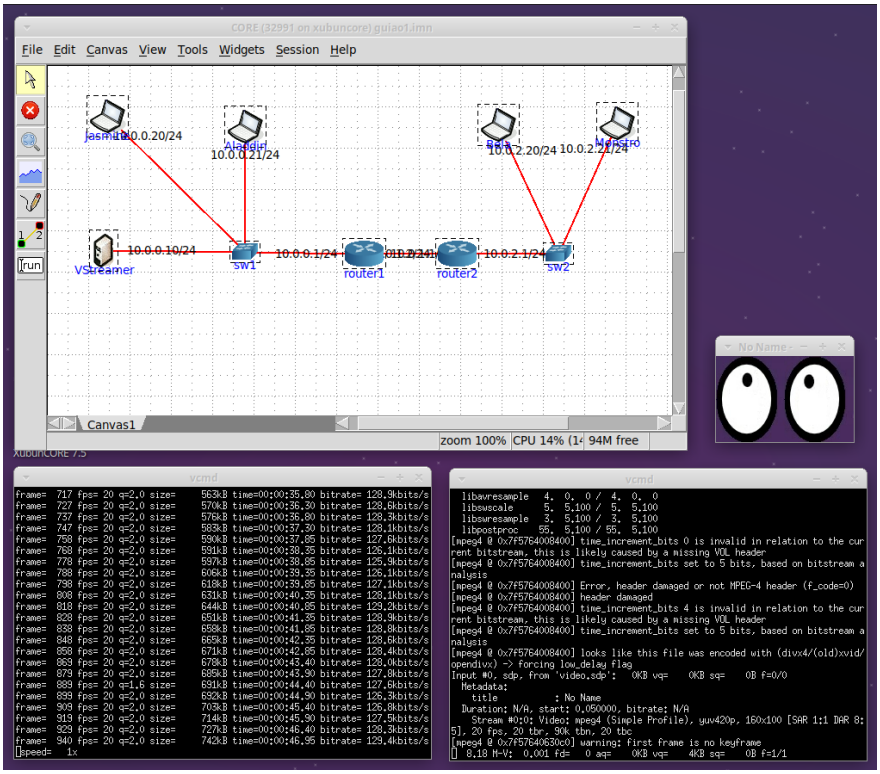


Figura 24 - Stream Unicast de VStreamer para Monstro

Passo 4

Capture o tráfego com o Wireshark no link de saída do servidor.

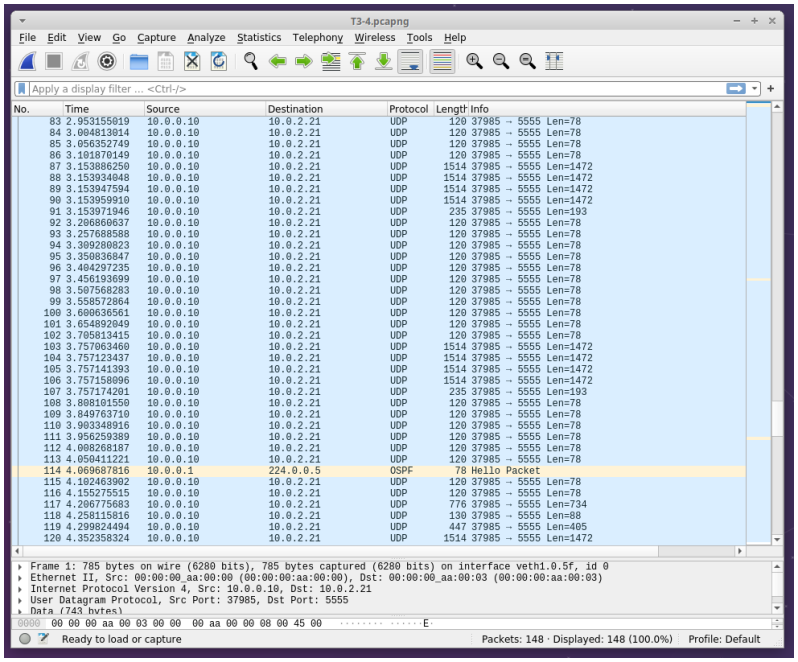


Figura 25 - Captura de wireshark na situação de unicast

Passo 6 e 7

Para o exercício multicast, vamos usar apenas um switch com o servidor e quatro portáteis ligados ao mesmo switch. Construa uma nova topologia como indicado na figura. Inicie a emulação CORE e teste a conectividade entre os sistemas.

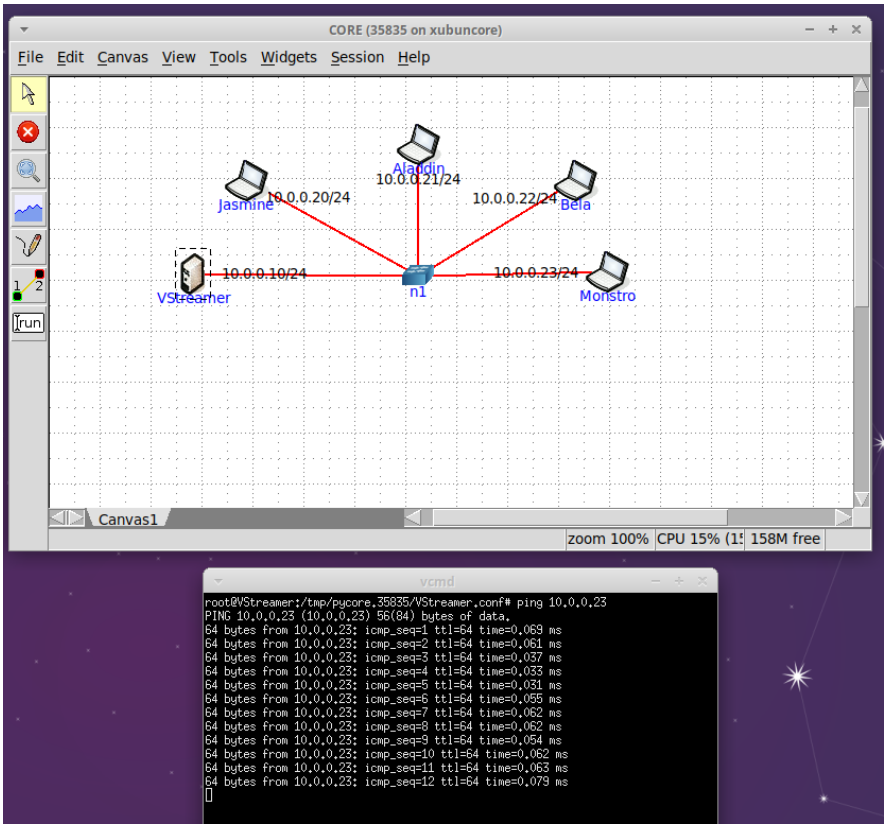


Figura 26 - Teste de conectividade da nova topologia

Passo 8 e 9

No servidor VStreamer, inicie uma sessão de streaming multicast com o ffmpeg. Em cada um dos portáteis (Jasmine, Aladdin, Bela e Monstro) inicie um cliente da sessão diferente

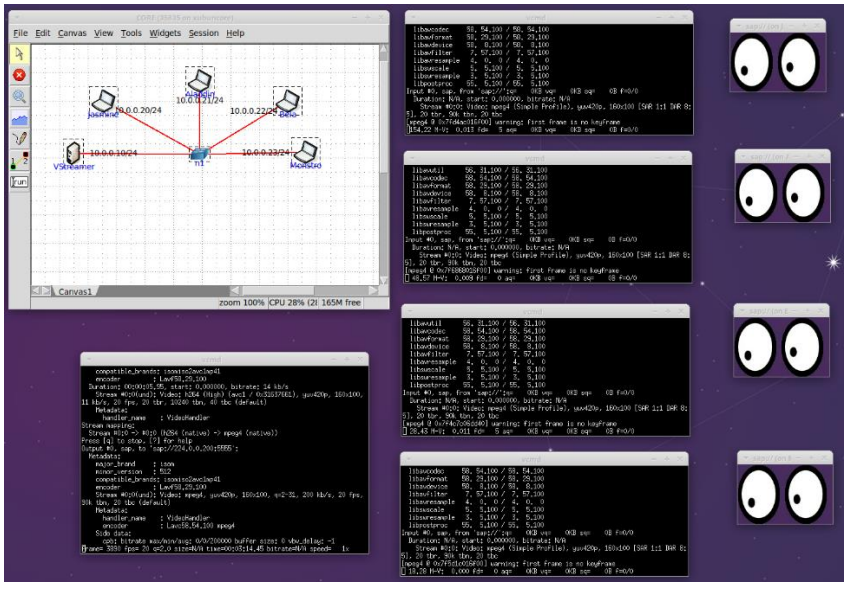


Figura 27 - Stream multicast com sessão aberta nos 4 portáteis

Passo 10

Capture o tráfego no link de saída do servidor com o Wireshark

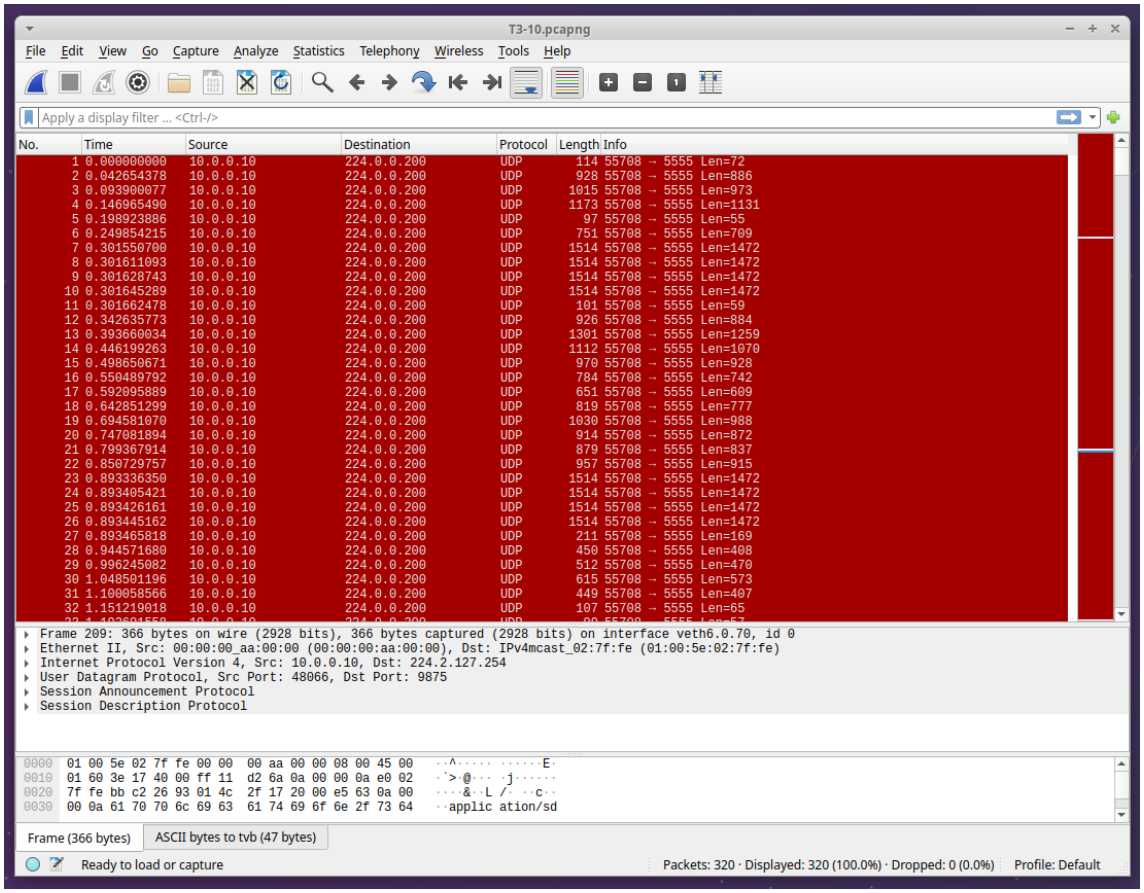


Figura 28 - Captura de wireshark na situação de Multicast

Questão 5

A utilização de unicast implica que existe um envio de pacotes de uma única fonte para um único destino. Isto implica a necessidade de saber o IP do destino para poder enviar estes mesmos pacotes. Além disso, na necessidade de enviar o mesmo pacote da fonte para diferentes destinos, teria de ser enviado 1 cópia para cada um desses, o que se revelaria num desperdício de largura de banda linear com o número de clientes para servir. Isto é verificável pelas capturas de Wireshark, visto que em unicast, existirá 2 conversações por cada destino (Figura 29,neste exemplo, o 'Monstro'). É neste sentido que o multicast vêm resolver estes problemas. O multicast consiste no envio de uma só cópia de dados para a rede, e a partir de cada switch/router vai existir uma distribuição desses dados pelas partes interessadas (Flooding) , quer sejam outros routers/switches ou hosts. Esta forma de transmissão permite resolver o problema de múltiplas cópias para diferentes clientes como o armazenamento de todos os IPs na fonte, que são substituídos por um único IP de broadcast (neste caso 224.0.0.200). No entanto, existem alguns pontos negativos desta forma de transmissão, como o overhead e complexidade de criar a árvore de distribuição

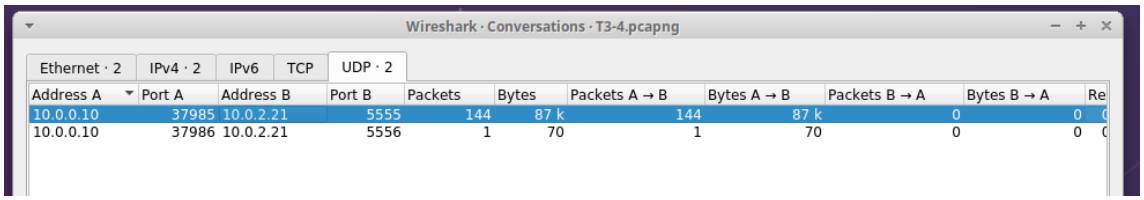


Figura 29 - Conversações geradas na situação de Unicast

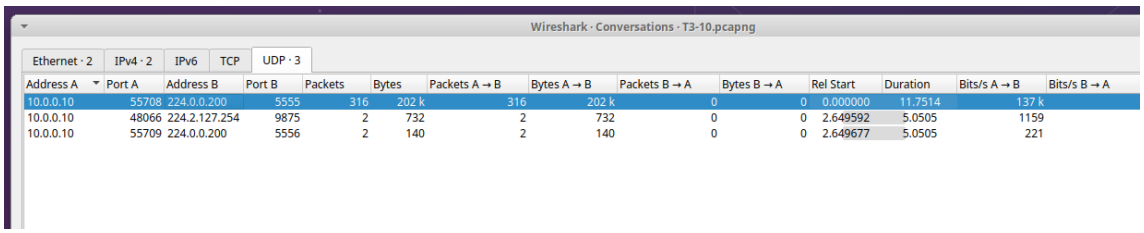


Figura 30 - Conversações geradas na situação de Multicast

Conclusões

No âmbito da unidade curricular de Engenharia de Serviços de Rede foi possível, através do Trabalho Prático 1, rever os conceitos aprendidos nas aulas teóricas na componente prática. No TP1 pudemos perceber as diferentes opções em termos de pilha protocolar e as diferenças conceptuais entre elas, realizando o streaming de áudio e vídeo a pedido em tempo real. Houve 3 etapas que continham objetivos de aprendizagem diferentes.

Na etapa 1, realizamos o streaming HTTP utilizando diferentes clientes , sendo estes : VLC, Firefox e ffplay , e assim foi possível verificar os diferentes níveis de débito consoante ao número de clientes.

Durante etapa 2, foi realizado o streaming adaptativo utizando o DASH(Dynamic Adaptive Streaming over HTTP) , onde foi possível criar o Manifest File , verificando as diferentes chunks para diferentes débitos, consoante a isso irá ser adaptavo aos clientes conforme a largura de banda que possuem.

Por fim, na etapa 3 foi colocado em prática o conceito de multicasting utilizando o SAP, o que nos fez perceber as diferenças entre o envio de pacotes em unicast em relação a multicast. Foi possível verificar os benefícios do endereçamento muticast , como retirar o peso de tráfego da fonte do servidor enviando apenas 1 pacote e sendo distribuído cópias dentro da topologia , independente do número de clientes.

Portanto , foi possível através deste trabalho prático aprofundar conceitos importantes no tema de Streaming de áudio e vídeo, e perceber as diferentes formas que pode ser enviado os pacotes conforme os clientes, além da familiarização das ferramentas de desenvolvimento, como Core e wireshark.