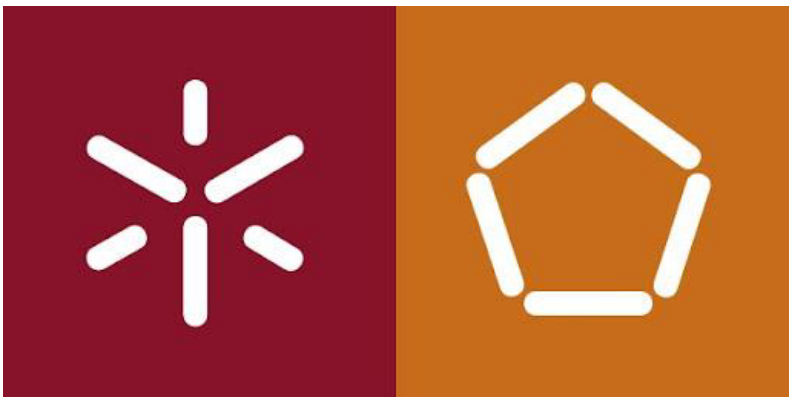
Universidade do Minho

Departamento de Informática
Mestrado em Engenharia Informática



Engenharia de Serviços em Rede

Relatório do TP1

Grupo nº43

Hugo dos Santos Martins
A95125

Ivo Miguel Alves Ribeiro
PG53886

João Bernardo Teixeira Escudeiro
A96075

11 de outubro de 2023

Índice:

Etapa 1. Streaming HTTP simples sem adaptação dinâmica de débito.

Topologia:

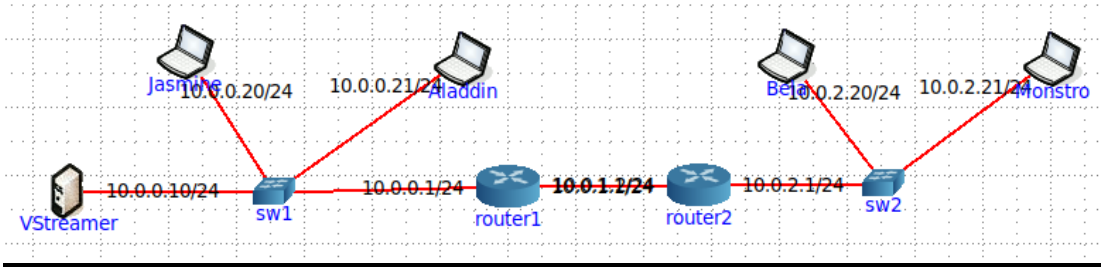


Figura 1 - Topologia utilizada para etapa 1.

Após criar a topologia foram realizados os seguintes comandos(*ping*) para testar a conectividade dos nós.

```
root@VStreamer:/tmp/pycore.39119/VStreamer.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=62 time=0.088 ms
64 bytes from 10.0.2.21: icmp_seq=2 ttl=62 time=0.047 ms
64 bytes from 10.0.2.21: icmp_seq=3 ttl=62 time=0.046 ms
64 bytes from 10.0.2.21: icmp_seq=4 ttl=62 time=0.047 ms
64 bytes from 10.0.2.21: icmp_seq=5 ttl=62 time=0.048 ms
```

Figura 2 - *Ping* de VStreamer para Monstro.

```
root@Jasmine:/tmp/pycore.39119/Jasmine.conf# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.078 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.045 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.046 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.057 ms
```

Figura 3 - *Ping* de Jasmine para Bela.

```
root@Aladdin:/tmp/pycore.39119/Aladdin.conf# traceroute 10.0.2.21
traceroute to 10.0.2.21 (10.0.2.21), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.052 ms  0.007 ms  0.005 ms
 2  10.0.1.2 (10.0.1.2)  0.014 ms  0.008 ms  0.007 ms
 3  10.0.2.21 (10.0.2.21)  0.070 ms  0.013 ms  0.011 ms
```

Figura 4 - *Ping* de Aladdin para Monstro.

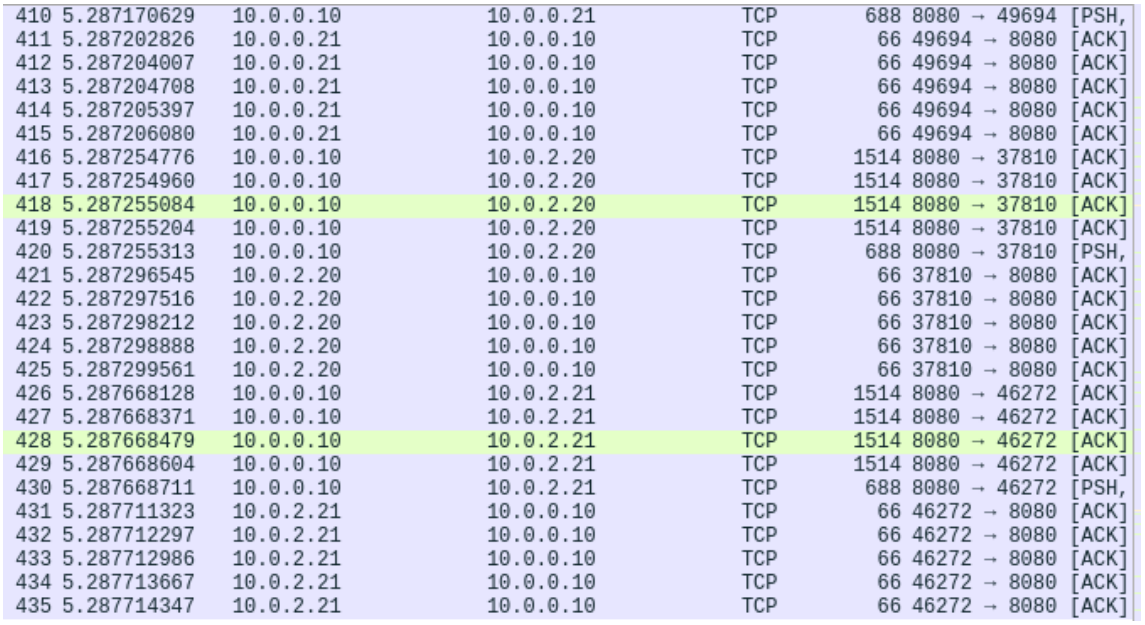
Após realizar todos os passos de 1.1 até 1.5, e verificar que o vídeo estava a ser transmitido, o grupo passou para o passo 1.6, em que o objetivo seria iniciar uma captura de tráfego utilizando a ferramenta *Wireshark*. A amostra de tráfego foi capturada na interface eth0 do servidor (10.0.0.10). Na figura 5 temos o resultado da captura apenas com um utilizador e nas figuras 6 e 7 com 2 e 3 utilizadores respetivamente.

355	35.465328114	10.0.0.10	10.0.0.20	TCP	278 8080 → 38390 [PSH,
356	35.465357614	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
357	35.465358806	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
358	35.465359483	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
359	35.465360165	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
360	36.014191173	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
361	36.464101935	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
362	36.464102422	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
363	36.464102556	10.0.0.10	10.0.0.20	TCP	1317 8080 → 38390 [PSH,
364	36.464128872	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
365	36.464130144	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
366	36.464130836	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
367	36.507007192	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
368	36.507007799	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
369	36.507007936	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
370	36.507008049	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
371	36.507008157	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [PSH,
372	36.507038923	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
373	36.507040157	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
374	36.507040838	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
375	36.507041502	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
376	36.507042170	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
377	36.507049282	10.0.0.10	10.0.0.20	TCP	1514 8080 → 38390 [ACK]
378	36.507049447	10.0.0.10	10.0.0.20	TCP	1193 8080 → 38390 [PSH,
379	36.507055414	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]
380	36.507056185	10.0.0.20	10.0.0.10	TCP	66 38390 → 8080 [ACK]

Figura 5 - Captura de tráfego no Servidor VStreamer (10.0.0.10) com comunicação para VLC (10.0.0.20).

484	14.453046645	10.0.0.21	10.0.0.10	TCP	66 49694 → 8080 [ACK]
485	14.453108265	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
486	14.453108499	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
487	14.453108643	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
488	14.453108765	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
489	14.453108872	10.0.0.10	10.0.2.20	TCP	688 8080 → 37808 [PSH,
490	14.453161985	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
491	14.453163014	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
492	14.453163690	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
493	14.453164358	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
494	14.453165030	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
495	15.070371546	00:00:00_aa:00:03	00:00:00_aa:00:00	ARP	42 Who has 10.0.0.10?
496	15.070382280	00:00:00_aa:00:00	00:00:00_aa:00:03	ARP	42 10.0.0.10 is at 00
497	15.084019571	10.0.0.10	10.0.0.21	TCP	1514 8080 → 49694 [ACK]
498	15.084019981	10.0.0.10	10.0.0.21	TCP	1514 8080 → 49694 [ACK]
499	15.084020094	10.0.0.10	10.0.0.21	TCP	1443 8080 → 49694 [PSH,
500	15.084046729	10.0.0.21	10.0.0.10	TCP	66 49694 → 8080 [ACK]
501	15.084048163	10.0.0.21	10.0.0.10	TCP	66 49694 → 8080 [ACK]
502	15.084048888	10.0.0.21	10.0.0.10	TCP	66 49694 → 8080 [ACK]
503	15.084108853	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
504	15.084109057	10.0.0.10	10.0.2.20	TCP	1514 8080 → 37808 [ACK]
505	15.084109192	10.0.0.10	10.0.2.20	TCP	1443 8080 → 37808 [PSH,
506	15.084149056	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
507	15.084149989	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]
508	15.084150666	10.0.2.20	10.0.0.10	TCP	66 37808 → 8080 [ACK]

Figura 6 - Captura de tráfego no Servidor VStreamer (10.0.0.10), com comunicação para dois clientes VLC (10.0.0.21) e Firefox (10.0.2.20).



410 5.287170629	10.0.0.10	10.0.0.21	TCP	688	8080 → 49694 [PSH,
411 5.287202826	10.0.0.21	10.0.0.10	TCP	66	49694 → 8080 [ACK]
412 5.287204007	10.0.0.21	10.0.0.10	TCP	66	49694 → 8080 [ACK]
413 5.287204708	10.0.0.21	10.0.0.10	TCP	66	49694 → 8080 [ACK]
414 5.287205397	10.0.0.21	10.0.0.10	TCP	66	49694 → 8080 [ACK]
415 5.287206080	10.0.0.21	10.0.0.10	TCP	66	49694 → 8080 [ACK]
416 5.287254776	10.0.0.10	10.0.2.20	TCP	1514	8080 → 37810 [ACK]
417 5.287254960	10.0.0.10	10.0.2.20	TCP	1514	8080 → 37810 [ACK]
418 5.287255084	10.0.0.10	10.0.2.20	TCP	1514	8080 → 37810 [ACK]
419 5.287255204	10.0.0.10	10.0.2.20	TCP	1514	8080 → 37810 [ACK]
420 5.287255313	10.0.0.10	10.0.2.20	TCP	688	8080 → 37810 [PSH,
421 5.287296545	10.0.2.20	10.0.0.10	TCP	66	37810 → 8080 [ACK]
422 5.287297516	10.0.2.20	10.0.0.10	TCP	66	37810 → 8080 [ACK]
423 5.287298212	10.0.2.20	10.0.0.10	TCP	66	37810 → 8080 [ACK]
424 5.287298888	10.0.2.20	10.0.0.10	TCP	66	37810 → 8080 [ACK]
425 5.287299561	10.0.2.20	10.0.0.10	TCP	66	37810 → 8080 [ACK]
426 5.287668128	10.0.0.10	10.0.2.21	TCP	1514	8080 → 46272 [ACK]
427 5.287668371	10.0.0.10	10.0.2.21	TCP	1514	8080 → 46272 [ACK]
428 5.287668479	10.0.0.10	10.0.2.21	TCP	1514	8080 → 46272 [ACK]
429 5.287668604	10.0.0.10	10.0.2.21	TCP	1514	8080 → 46272 [ACK]
430 5.287668711	10.0.0.10	10.0.2.21	TCP	688	8080 → 46272 [PSH,
431 5.287711323	10.0.2.21	10.0.0.10	TCP	66	46272 → 8080 [ACK]
432 5.287712297	10.0.2.21	10.0.0.10	TCP	66	46272 → 8080 [ACK]
433 5.287712986	10.0.2.21	10.0.0.10	TCP	66	46272 → 8080 [ACK]
434 5.287713667	10.0.2.21	10.0.0.10	TCP	66	46272 → 8080 [ACK]
435 5.287714347	10.0.2.21	10.0.0.10	TCP	66	46272 → 8080 [ACK]

Figura 7 - Captura de tráfego no Servidor VStreamer (10.0.0.10),
com comunicação para três clientes VLC (10.0.0.21),
Firefox (10.0.2.20) e FFplay (10.0.2.21).

```
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoA.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.29.100
  Duration: 00:00:07.85, start: 0.000000, bitrate: 18 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 200x150,
16 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
    Metadata:
      handler_name    : VideoHandler
```

Figura 8 - Informação do vídeo obtido através do comando **ffprobe videoA.mp4**.

```
▸ Frame 361: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth1.0.57, id 0
▾ Ethernet II, Src: 00:00:00_aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00_aa:00:01 (00:00:00:aa:00:01)
    ▸ Destination: 00:00:00_aa:00:01 (00:00:00:aa:00:01)
    ▸ Source: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
      Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20
▸ Transmission Control Protocol, Src Port: 8080, Dst Port: 38390, Seq: 201943, Ack: 135, Len: 1448
```

Figura 9 - Informação do encapsulamento de um pacote.

Questão 1:

- Capture três pequenas amostras de tráfego no link de saída do servidor, respetivamente com 1 cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay). Identifique a taxa em bps necessária (usando o ffmpeg -i videoA.mp4 e/ou o próprio wireshark), o encapsulamento usado e o número total de fluxos gerados. Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício (ex: capturas de ecrã).

Através das capturas presentes nas figuras 5,6 e 7, conseguimos observar o fluxo de tráfego utilizando um diferente número de clientes.

Observando a figura 8, conseguimos perceber que a largura de banda necessária para a transmissão do vídeo em bps é 18 kb/s.

No pacote da figura 9, conseguimos perceber que foi utilizado o protocolo Ethernet na camada de ligação e o protocolo IPV4 na camada de rede.

Na figura 5 está evidente o tráfego através do protocolo TCP entre o fornecedor do vídeo-Servidor VStreamer (IP 10.0.0.10) e um único cliente-VLC (10.0.0.20). Neste caso apenas existe um fluxo, visto que os IPS e as entidades envolvidas nas trocas são apenas duas (servidor e *host*) figura 10.

Observando a figura 6, conseguimos perceber que nesse caso existiram 2 clientes, o VLC (10.0.0.21) e o Firefox (10.0.2.20). Para este caso e visto que existem dois clientes, então o número de fluxos é dois (figura 11).

Por último, e analisando detalhadamente a figura 7, conseguimos observar 4 IPS diferentes. O IP do Servidor VStreamer (10.0.0.10), e o IP dos 3 clientes VLC (10.0.0.21), Firefox (10.0.2.20) e FFPlay (10.0.2.21). Neste caso existem 3 fluxos figura 12.

Após retiradas as conclusões conseguimos perceber que o número de fluxos é diretamente proporcional ao número de clientes. Prevê-se que a taxa de *bitrate* cresça conforme o número de *streams*, o que nos leva a um problema de escalabilidade quando existir um número elevado de clientes,

De notar que no primeiro fluxo o grupo usou o vlc no *host* Jasmine e nos dois seguintes usou no Aladin.

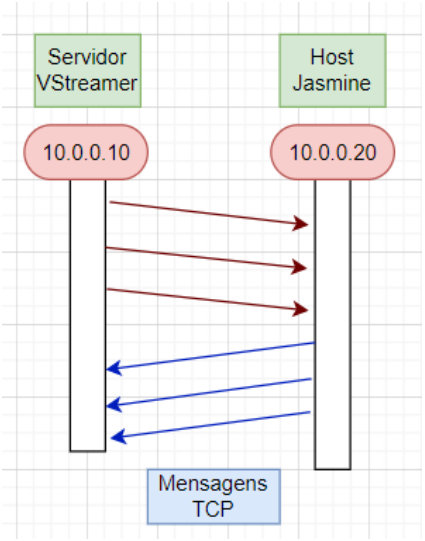


Figura 10 - Representação de 1 Fluxo VStreamer→ Jasmine.

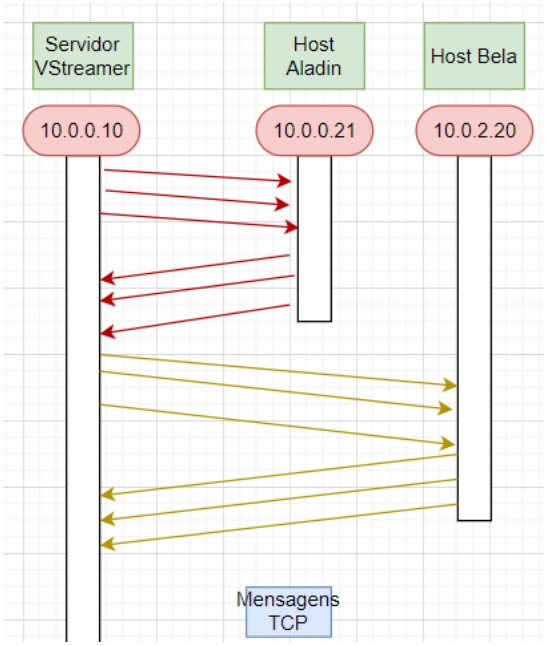


Figura 11 - Representação de 2 Fluxos VStreamer→ Aladin(Vermelho) e VStreamer→ Bela (Amarelo).

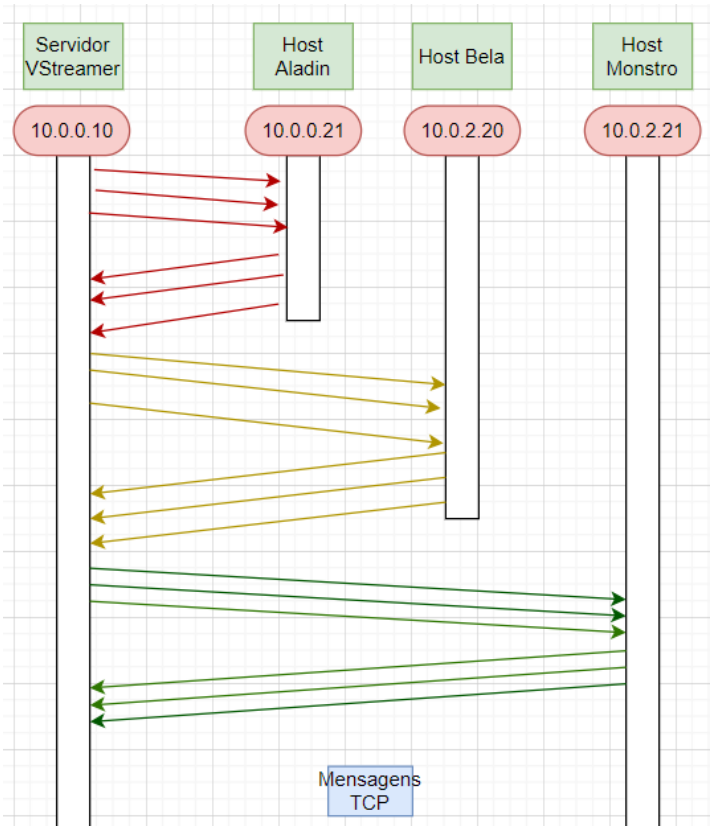


Figura 12 - Representação de 3 Fluxos VStreamer→ Aladin(Vermelho) ,VStreamer→ Bela (Amarelo) e VStreamer→ Monstro (Verde) .

Etapa 2. Streaming adaptativo sobre HTTP (MPEG-DASH).

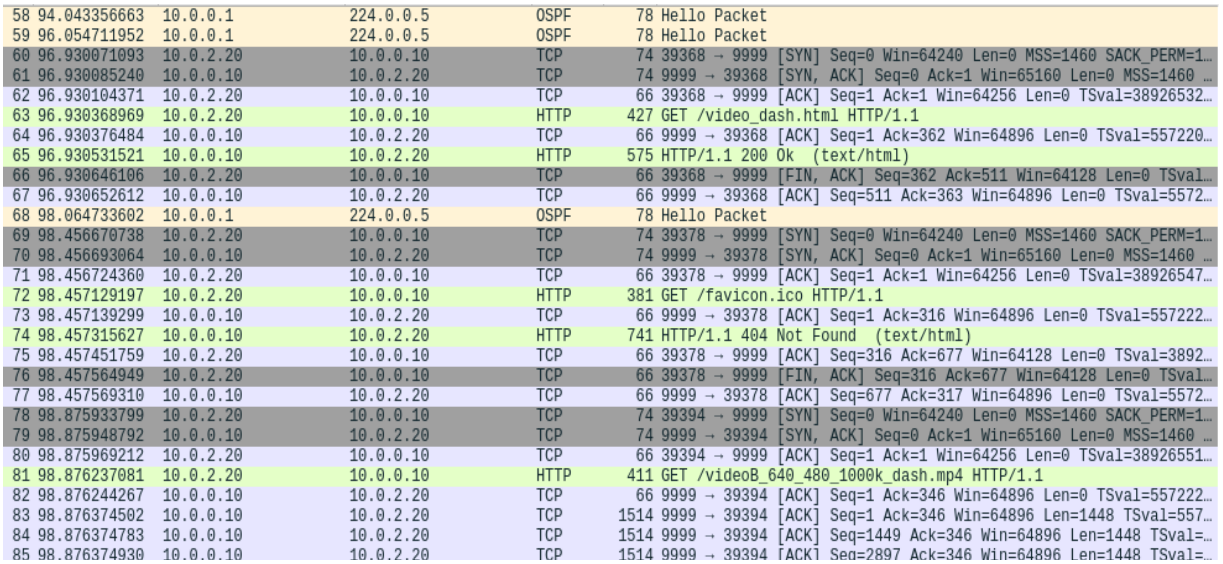
Após concluir todas as etapas da etapa 1, o grupo passou então à realização da tarefa 2, que consistia na observação do *streaming* com débito adaptativo. Assim, foram realizados todos os passos de 1 até 10, com as respetivas criações dos novos vídeos e ficheiros auxiliares.

O grupo realizou o comando ***ffprobe*** para os 3 vídeos para perceber a taxa de *bitrate* necessária para a transmissão de cada um dos vídeos de diferentes qualidades.

Figura 13 - ffprobe videoB_200_150_200K.mp4.

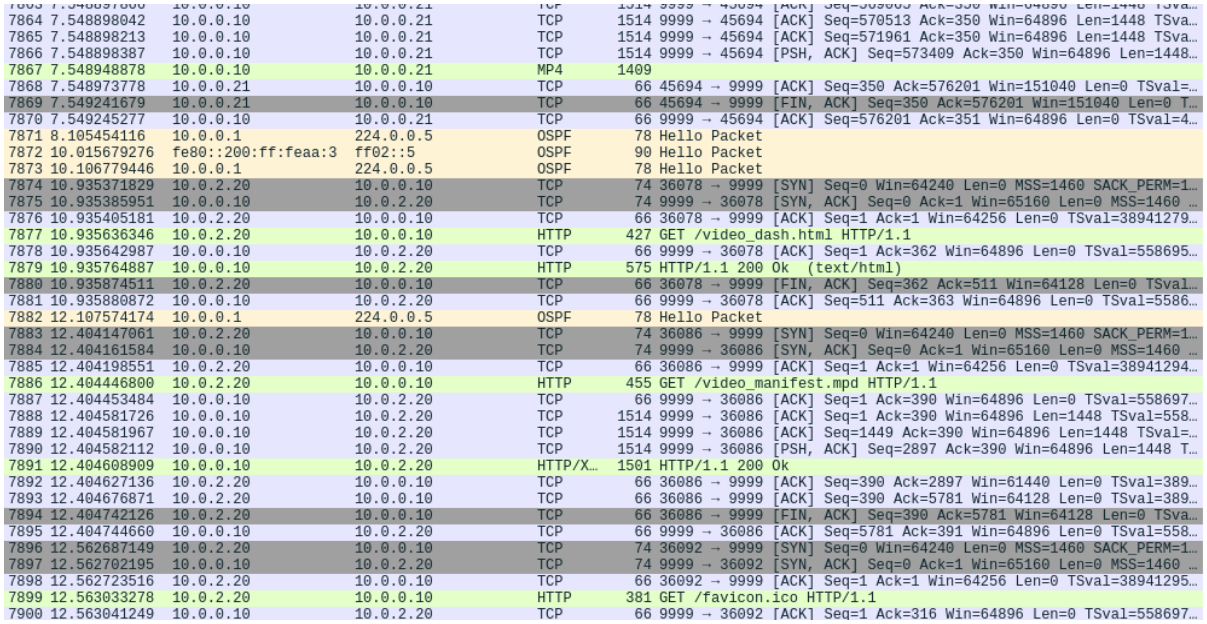
Figura 14 - ffprobe videoB_480_360_500K.mp4.

Figura 15 - ffprobe videoB_640_480_1000K.mp4.



58	94.043356663	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
59	96.054711952	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
60	96.930071093	10.0.2.20	10.0.0.10	TCP	74 39368 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
61	96.930085240	10.0.0.10	10.0.2.20	TCP	74 9999 → 39368 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
62	96.930104371	10.0.2.20	10.0.0.10	TCP	66 39368 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38926532…
63	96.930368969	10.0.2.20	10.0.0.10	HTTP	427 GET /video_dash.html HTTP/1.1
64	96.930376484	10.0.0.10	10.0.2.20	TCP	66 9999 → 39368 [ACK] Seq=1 Ack=362 Win=64896 Len=0 TSval=557220…
65	96.930531521	10.0.0.10	10.0.2.20	HTTP	575 HTTP/1.1 200 Ok (text/html)
66	96.930646106	10.0.2.20	10.0.0.10	TCP	66 39368 → 9999 [FIN, ACK] Seq=362 Ack=511 Win=64128 Len=0 TSval…
67	96.930652612	10.0.0.10	10.0.2.20	TCP	66 9999 → 39368 [ACK] Seq=511 Ack=363 Win=64896 Len=0 TSval=5572…
68	98.064733602	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
69	98.456670738	10.0.2.20	10.0.0.10	TCP	74 39378 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
70	98.456693064	10.0.0.10	10.0.2.20	TCP	74 9999 → 39378 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
71	98.456724360	10.0.2.20	10.0.0.10	TCP	66 39378 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38926547…
72	98.457129197	10.0.2.20	10.0.0.10	HTTP	381 GET /favicon.ico HTTP/1.1
73	98.457139299	10.0.0.10	10.0.2.20	TCP	66 9999 → 39378 [ACK] Seq=1 Ack=316 Win=64896 Len=0 TSval=557222…
74	98.457315627	10.0.0.10	10.0.2.20	HTTP	741 HTTP/1.1 404 Not Found (text/html)
75	98.457451759	10.0.2.20	10.0.0.10	TCP	66 39378 → 9999 [ACK] Seq=316 Ack=677 Win=64128 Len=0 TSval=3892…
76	98.457564949	10.0.2.20	10.0.0.10	TCP	66 39378 → 9999 [FIN, ACK] Seq=316 Ack=677 Win=64128 Len=0 TSval…
77	98.457569310	10.0.0.10	10.0.2.20	TCP	66 9999 → 39378 [ACK] Seq=677 Ack=317 Win=64896 Len=0 TSval=5572…
78	98.875933799	10.0.2.20	10.0.0.10	TCP	74 39394 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
79	98.875948792	10.0.0.10	10.0.2.20	TCP	74 9999 → 39394 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
80	98.875969212	10.0.2.20	10.0.0.10	TCP	66 39394 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38926551…
81	98.876237081	10.0.2.20	10.0.0.10	HTTP	411 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
82	98.876244267	10.0.0.10	10.0.2.20	TCP	66 9999 → 39394 [ACK] Seq=1 Ack=346 Win=64896 Len=0 TSval=557222…
83	98.876374502	10.0.0.10	10.0.2.20	TCP	1514 9999 → 39394 [ACK] Seq=1 Ack=346 Win=64896 Len=1448 TSval=557…
84	98.876374783	10.0.0.10	10.0.2.20	TCP	1514 9999 → 39394 [ACK] Seq=1449 Ack=346 Win=64896 Len=1448 TSval=…
85	98.876374930	10.0.0.10	10.0.2.20	TCP	1514 9999 → 39394 [ACK] Seq=2897 Ack=346 Win=64896 Len=1448 TSval=…

Figura 16 - Captura da alínea 9), VStreamer a transmitir e Bella a receber através do Firefox.



7864	7.548898042	10.0.0.10	10.0.0.21	TCP	1514 9999 → 45694 [ACK] Seq=570513 Ack=350 Win=64896 Len=1448 TSva…
7865	7.548898213	10.0.0.10	10.0.0.21	TCP	1514 9999 → 45694 [ACK] Seq=571961 Ack=350 Win=64896 Len=1448 TSva…
7866	7.548898387	10.0.0.10	10.0.0.21	TCP	1514 9999 → 45694 [PSH, ACK] Seq=573409 Ack=350 Win=64896 Len=1448…
7867	7.548948878	10.0.0.10	10.0.0.21	MP4	1409
7868	7.548973778	10.0.0.21	10.0.0.10	TCP	66 45694 → 9999 [ACK] Seq=350 Ack=576201 Win=151040 Len=0 TSval=…
7869	7.549241679	10.0.0.21	10.0.0.10	TCP	66 45694 → 9999 [FIN, ACK] Seq=350 Ack=576201 Win=151040 Len=0 T…
7870	7.549245277	10.0.0.10	10.0.0.21	TCP	66 9999 → 45694 [ACK] Seq=576201 Ack=351 Win=64896 Len=0 TSval=4…
7871	8.105454116	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
7872	10.015679276	fe80::200:ff:feaa:3	ff02::5	OSPF	90 Hello Packet
7873	10.106779446	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
7874	10.935371829	10.0.2.20	10.0.0.10	TCP	74 36078 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
7875	10.935385951	10.0.0.10	10.0.2.20	TCP	74 9999 → 36078 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
7876	10.935405181	10.0.2.20	10.0.0.10	TCP	66 36078 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38941279…
7877	10.935636346	10.0.2.20	10.0.0.10	HTTP	427 GET /video_dash.html HTTP/1.1
7878	10.935642987	10.0.0.10	10.0.2.20	TCP	66 9999 → 36078 [ACK] Seq=1 Ack=362 Win=64896 Len=0 TSval=558695…
7879	10.935764887	10.0.0.10	10.0.2.20	HTTP	575 HTTP/1.1 200 Ok (text/html)
7880	10.935874511	10.0.2.20	10.0.0.10	TCP	66 36078 → 9999 [FIN, ACK] Seq=362 Ack=511 Win=64128 Len=0 TSval…
7881	10.935880872	10.0.0.10	10.0.2.20	TCP	66 9999 → 36078 [ACK] Seq=511 Ack=363 Win=64896 Len=0 TSval=5586…
7882	12.107574174	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
7883	12.404147061	10.0.2.20	10.0.0.10	TCP	74 36086 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
7884	12.404161584	10.0.0.10	10.0.2.20	TCP	74 9999 → 36086 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
7885	12.404198551	10.0.2.20	10.0.0.10	TCP	66 36086 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38941294…
7886	12.404446800	10.0.2.20	10.0.0.10	HTTP	455 GET /video_manifest.mpd HTTP/1.1
7887	12.404453484	10.0.0.10	10.0.2.20	TCP	66 9999 → 36086 [ACK] Seq=1 Ack=390 Win=64896 Len=0 TSval=558697…
7888	12.404581726	10.0.0.10	10.0.2.20	TCP	1514 9999 → 36086 [ACK] Seq=1 Ack=390 Win=64896 Len=1448 TSval=558…
7889	12.404581967	10.0.0.10	10.0.2.20	TCP	1514 9999 → 36086 [ACK] Seq=1449 Ack=390 Win=64896 Len=1448 TSval=…
7890	12.404582112	10.0.0.10	10.0.2.20	TCP	1514 9999 → 36086 [PSH, ACK] Seq=2897 Ack=390 Win=64896 Len=1448 T…
7891	12.404608909	10.0.0.10	10.0.2.20	HTTP/X…	1501 HTTP/1.1 200 Ok
7892	12.404627136	10.0.2.20	10.0.0.10	TCP	66 36086 → 9999 [ACK] Seq=390 Ack=2897 Win=61440 Len=0 TSval=389…
7893	12.404676871	10.0.2.20	10.0.0.10	TCP	66 36086 → 9999 [ACK] Seq=390 Ack=5781 Win=64128 Len=0 TSval=389…
7894	12.404742126	10.0.2.20	10.0.0.10	TCP	66 36086 → 9999 [FIN, ACK] Seq=390 Ack=5781 Win=64128 Len=0 TSva…
7895	12.404744660	10.0.0.10	10.0.2.20	TCP	66 9999 → 36086 [ACK] Seq=5781 Ack=391 Win=64896 Len=0 TSval=558…
7896	12.562687149	10.0.2.20	10.0.0.10	TCP	74 36092 → 9999 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
7897	12.562702195	10.0.0.10	10.0.2.20	TCP	74 9999 → 36092 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
7898	12.562723516	10.0.2.20	10.0.0.10	TCP	66 36092 → 9999 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=38941295…
7899	12.563033278	10.0.2.20	10.0.0.10	HTTP	381 GET /favicon.ico HTTP/1.1
7900	12.563041249	10.0.0.10	10.0.2.20	TCP	66 9999 → 36092 [ACK] Seq=1 Ack=316 Win=64896 Len=0 TSval=558697…

Figura 17 - Captura da alínea 10), VStreamer a transmitir,
Bella e Aladin a receber através do Firefox.

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.

Para que ocorra transmissão do vídeo B, irá ser necessária uma largura de banda de pelo menos 113 kb/s (figura 13), que corresponde à versão de vídeo com menor qualidade, e para que seja transmitido o vídeo de maior qualidade será necessária uma taxa de 517 kb/s (figura 15).

Para analisar a pilha protocolar, o grupo utilizou uma trama *Wireshark* (Figura 18). Na figura 19 está um pequeno esboço da pilha protocolar utilizada.

```
13 2.378327541   10.0.2.20    10.0.0.10    TCP      66 38366 → 9991 [ACK] Seq=1 Ac
14 2.378601536   10.0.2.20    10.0.0.10    HTTP    365 GET /dash.all.debug.js HTTP
15 2.378609329   10.0.0.10    10.0.2.20    TCP      66 9991 → 38366 [ACK] Seq=1 Ac
16 2.433998876   10.0.0.10    10.0.2.20    TCP    1514 9991 → 38366 [ACK] Seq=1 Ac
17 2.433999549   10.0.0.10    10.0.2.20    TCP    1514 9991 → 38366 [ACK] Seq=1449
18 2.433999739   10.0.0.10    10.0.2.20    TCP    1514 9991 → 38366 [ACK] Seq=2897
Frame 6: 477 bytes on wire (3816 bits), 477 bytes captured (3816 bits) on interface veth1.0.b8, id 0
Ethernet II, Src: 00:00:00_aa:00:03 (00:00:00:aa:00:03), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.0.10
Transmission Control Protocol, Src Port: 38356, Dst Port: 9991, Seq: 1, Ack: 1, Len: 411
Hypertext Transfer Protocol
  GET /video_dash.html HTTP/1.1\r\n
  Host: 10.0.0.10:9991\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  If-Modified-Since: Mon, 09 Oct 2023 09:59:49 GMT\r\n
  \r\n
  [Full request URI: http://10.0.0.10:9991/video_dash.html]
  [HTTP request 1/1]
  [Response in frame: 8]
```

Figura 18 - Pacote HTTP.

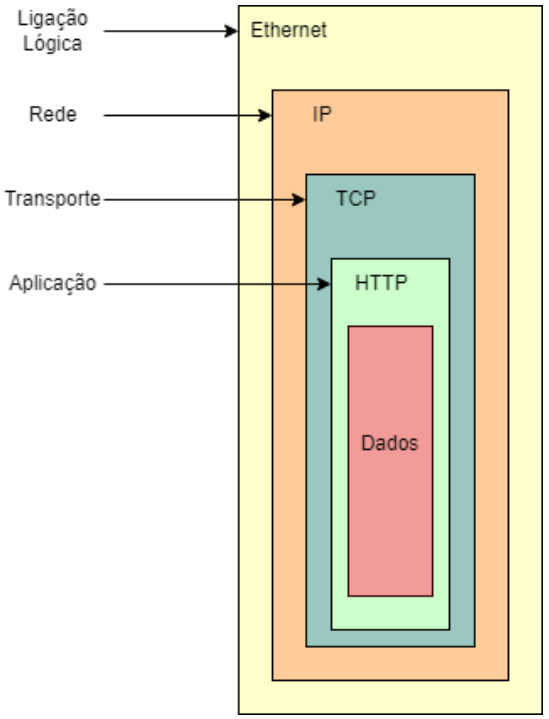


Figura 19 - Pilha protocolar utilizada.

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.

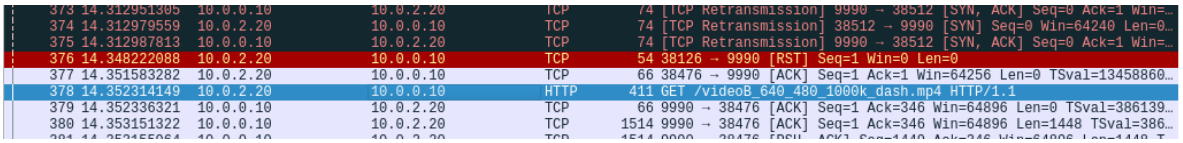


Figura 20 - HTTP Get da Bela.

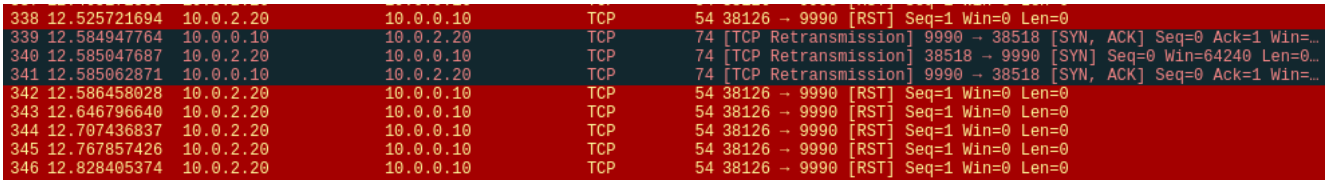


Figura 21 - Erros e redução de qualidade em Bela.

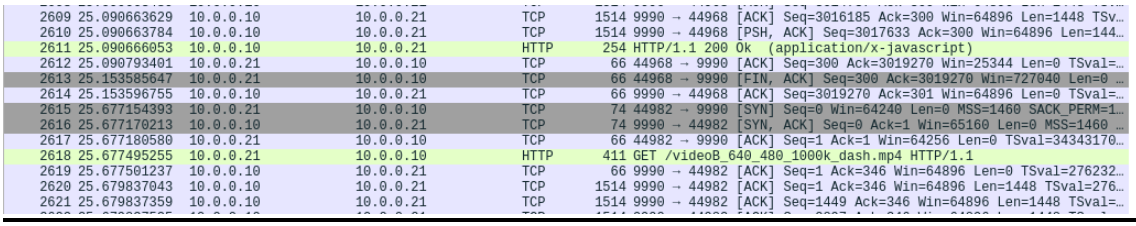


Figura 22 - HTTP Get do Aladdin do vídeo de maior qualidade.

Nas capturas das figuras 20,21 e 22, salientam-se os seguintes pontos:

◦ É expectável que no portátil Bela haja uma solicitação do vídeo de maior qualidade. O mesmo não será transmitido devido aos *timeouts* que ocorrem. Estes *timeouts* devem-se ao facto de o tempo de *timeout* definido pelo protocolo utilizado ser excedido na transmissão. A largura de banda disponível é um causador do timeout, visto que não será possível transmitir o vídeo na qualidade inicial, no tempo definido figura 21.

◦De seguida, seria expectável que existisse um GET da Bela do vídeo de qualidade intermédia, pelo que não apareceu na captura de tráfego realizada, mesmo realizando 3 capturas diferentes.

◦Por fim ocorre um GET da bela do vídeo de menor qualidade visto que não foi possível transmitir o vídeo nas qualidades superiores. Esta transmissão ocorrerá sem erros, visto que a largura de banda disponível permite que a transmissão ocorra.

O grupo decidiu limitar o link de ligação do sw2 (*switch* 2) e o portátil Bela, para um débito máximo de 200 kb/s, com o objetivo de forçar a mudança do vídeo, visto que um valor intermédio de *bitrate* quando comparados os valores de *bitrate* do video de menor e intermedia qualidade.

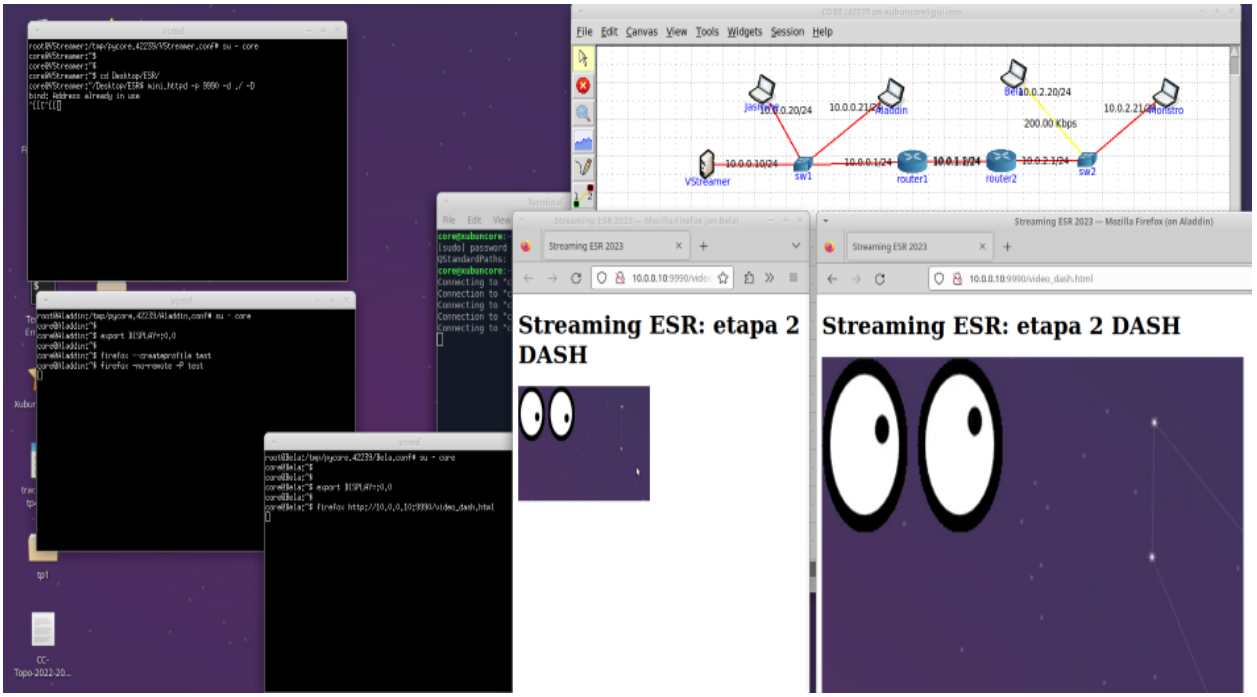


Figura 23 - Interface de utilização com todos os vídeos a serem transmitidos.

Questão 4:

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

Através do DASH, o ficheiro de vídeo será codificado em diferentes versões, cada uma com qualidade diferente, e também com uma largura de banda diferente. Os "pedaços" de vídeo são solicitados pelo cliente utilizando de pedidos HTTPGET. O cliente "escolhe" a qualidade de vídeo mediante a largura de banda disponível. O protocolo DASH é útil na medida em que permite que haja uma oscilação do vídeo que é transferido pela rede, o que permite ao cliente solicitar diferentes níveis de qualidade.

```
-<MPD minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H0M8.867S" 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="PT0H0M8.867S">
  -<AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="640" maxHeight="480" maxFrameRate="30" par="4:3" lang="und">
   -<SegmentList>
      <Initialization sourceURL="video_manifest_init.mp4"/>
    </SegmentList>
   -<Representation id="1" mimeType="video/mp4" codecs="avc3.64000c" width="200" height="150" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="115736">
      <BaseURL>videoB_200_150_200k_dash.mp4</BaseURL>
     +<SegmentList timescale="15360" duration="7680"></SegmentList>
    </Representation>
   -<Representation id="2" mimeType="video/mp4" codecs="avc3.64001e" width="480" height="360" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="331249">
      <BaseURL>videoB_480_360_500k_dash.mp4</BaseURL>
     +<SegmentList timescale="15360" duration="7680"></SegmentList>
    </Representation>
   -<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="480" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="519693">
      <BaseURL>videoB_640_480_1000k_dash.mp4</BaseURL>
     +<SegmentList timescale="15360" duration="7680"></SegmentList>
    </Representation>
  </AdaptationSet>
 </Period>
</MPD>
```

Figura 24- Ficheiro MPD utilizado.

Etapa 3. Streaming RTP/RTCP unicast sobre UDP e multicast com anúncios SAP.

O objetivo da etapa 3 seria, maioritariamente, comparar em termos de eficiência e largura de banda utilizando dois cenários de *streaming*: *unicast* ou *multicast*. Assim, e após realizar todas as tarefas propostas surgem as seguintes figuras que comprovam que as etapas foram seguidas meticulosamente. O grupo começou por desenvolver uma topologia apenas com um *switch*, com um servidor e 4 portáteis.

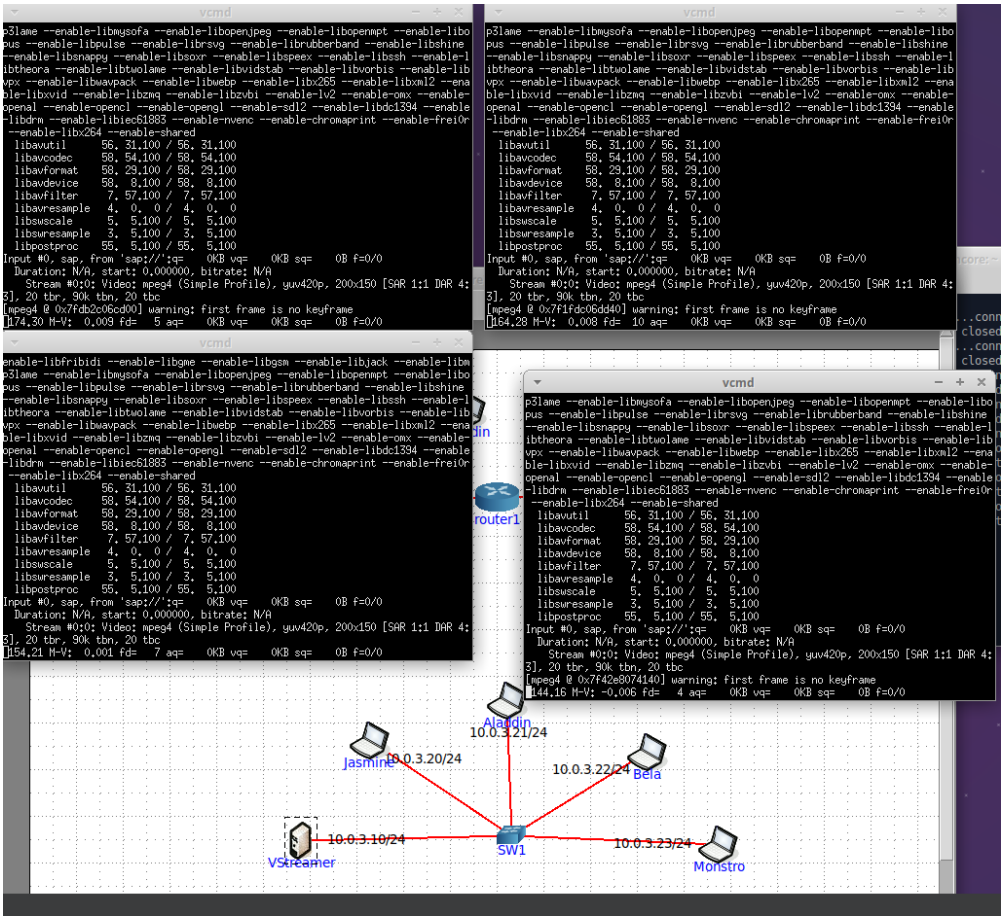


Figura 25 - Transmissão para 4 Portáteis (Bella, Aladin, Jasmine, Monstro).

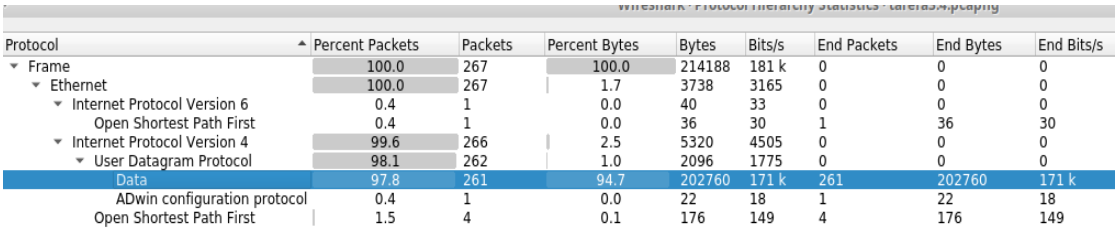
Figura 26 -Assistir as Streams Simultaneamente em 4 portáteis (Bella, Aladin, Jasmine, Monstro).

421	15.164203503	10.0.3.10	224.0.0.200	MP4V-ES	1152	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6310, Time=3561460480, Mark
422	15.199740158	10.0.3.10	224.0.0.200	MP4V-ES	834	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6311, Time=3561464980, Mark
423	15.251535330	10.0.3.10	224.0.0.200	MP4V-ES	1181	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6312, Time=3561469480, Mark
424	15.302893584	10.0.3.10	224.0.0.200	MP4V-ES	130	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6313, Time=3561473980, Mark
425	15.354575108	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6314, Time=3561478480
426	15.354629554	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6315, Time=3561478480
427	15.354641540	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6316, Time=3561478480
428	15.354652443	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6317, Time=3561478480
429	15.354664910	10.0.3.10	224.0.0.200	MP4V-ES	409	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6318, Time=3561478480, Mark
430	15.407345772	10.0.3.10	224.0.0.200	MP4V-ES	1188	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6319, Time=3561482980, Mark
431	15.459363376	10.0.3.10	224.0.0.200	MP4V-ES	1182	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6320, Time=3561487480, Mark
432	15.501321228	10.0.3.10	224.0.0.200	MP4V-ES	1351	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6321, Time=3561491980, Mark
433	15.554544669	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6322, Time=3561496480
434	15.554620682	10.0.3.10	224.0.0.200	MP4V-ES	86	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6323, Time=3561496480, Mark
435	15.606872377	10.0.3.10	224.0.0.200	MP4V-ES	345	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6324, Time=3561500980, Mark
436	15.651813936	10.0.3.10	224.0.0.200	MP4V-ES	1157	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6325, Time=3561505480, Mark
437	15.713619713	10.0.3.10	224.0.0.200	MP4V-ES	1331	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6326, Time=3561509980, Mark
438	15.755339365	10.0.3.10	224.0.0.200	MP4V-ES	396	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6327, Time=3561514480, Mark
439	15.801438374	10.0.3.10	224.0.0.200	MP4V-ES	122	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6328, Time=3561518980, Mark
440	15.854170036	10.0.3.10	224.0.0.200	MP4V-ES	1338	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6329, Time=3561523480, Mark
441	15.899702307	10.0.3.10	224.0.0.200	MP4V-ES	263	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6330, Time=3561527980, Mark
442	15.952672764	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6331, Time=3561532480
443	15.954365195	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6332, Time=3561532480
444	15.954422573	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6333, Time=3561532480
445	15.954443159	10.0.3.10	224.0.0.200	MP4V-ES	1514	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6334, Time=3561532480
446	15.954465208	10.0.3.10	224.0.0.200	MP4V-ES	371	PT=MP4V-ES, SSRC=0x5BB48657, Seq=6335, Time=3561532480, Mark

Figura 27 -Captura Wireshark durante a transmissão para os 4 portáteis.

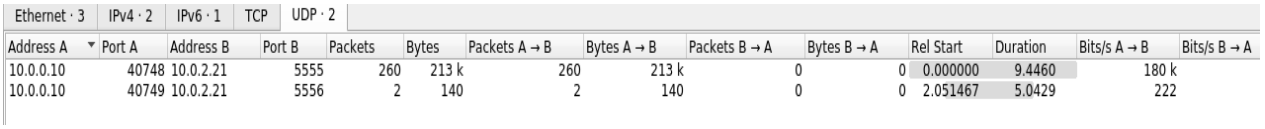
Questão 5:

- Compare o cenário unicast aplicado com o cenário multicast. Mostre vantagens e desvantagens na solução multicast ao nível da rede, no que diz respeito a escalabilidade (aumento do nº de clientes) e tráfego na rede. Tire as suas conclusões.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	267	100.0	214188	181 k	0	0	0
Ethernet	100.0	267	1.7	3738	3165	0	0	0
Internet Protocol Version 6	0.4	1	0.0	40	33	0	0	0
Open Shortest Path First	0.4	1	0.0	36	30	1	36	30
Internet Protocol Version 4	99.6	266	2.5	5320	4505	0	0	0
User Datagram Protocol	98.1	262	1.0	2096	1775	0	0	0
Data	97.8	261	94.7	202760	171 k	261	202760	171 k
ADwin configuration protocol	0.4	1	0.0	22	18	1	22	18
Open Shortest Path First	1.5	4	0.1	176	149	4	176	149

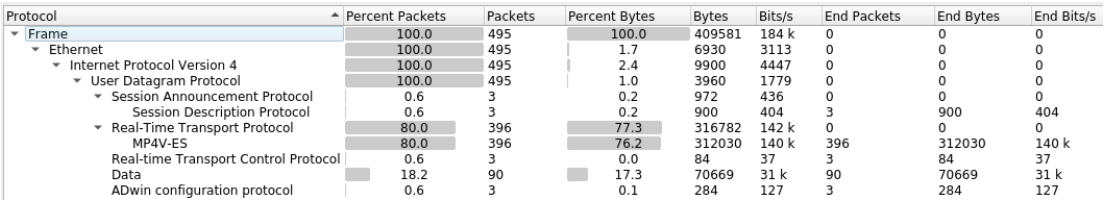
Figura 28 - Captura Estatística *Hierarchy* no cenário *unicast*.



Ethernet · 3 | IPv4 · 2 | IPv6 · 1 | TCP | UDP · 2

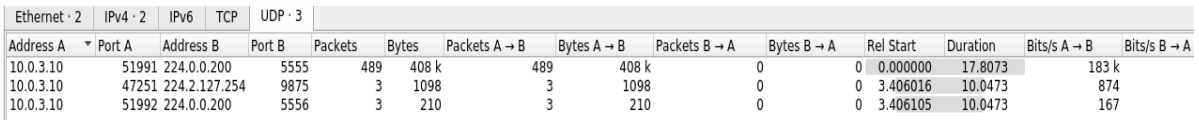
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.10	40748	10.0.2.21	5555	260	213 k	260	213 k	0	0	0.000000	9.4460	180 k	
10.0.0.10	40749	10.0.2.21	5556	2	140	2	140	0	0	2.051467	5.0429	222	

Figura 29 - Captura Estatística *Conversations* no cenário *unicast*.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	495	100.0	409581	184 k	0	0	0
Ethernet	100.0	495	1.7	6930	3113	0	0	0
Internet Protocol Version 4	100.0	495	2.4	9900	4447	0	0	0
User Datagram Protocol	100.0	495	1.0	3960	1779	0	0	0
Session Announcement Protocol	0.6	3	0.2	972	436	0	0	0
Session Description Protocol	0.6	3	0.2	900	404	3	900	404
Real-Time Transport Protocol	80.0	396	77.3	316782	142 k	0	0	0
MP4V-ES	80.0	396	76.2	312030	140 k	396	312030	140 k
Real-time Transport Control Protocol	0.6	3	0.0	84	37	3	84	37
Data	18.2	90	17.3	70669	31 k	90	70669	31 k
ADwin configuration protocol	0.6	3	0.1	284	127	3	284	127

Figura 30 - Captura Estatística *Hierarchy* no cenário *multicast*



Ethernet · 2 | IPv4 · 2 | IPv6 | TCP | UDP · 3

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.10	51991	224.0.0.200	5555	489	408 k	489	408 k	0	0	0.000000	17.8073	183 k	
10.0.3.10	47251	224.2.127.254	9875	3	1098	3	1098	0	0	3.406016	10.0473	874	
10.0.3.10	51992	224.0.0.200	5556	3	210	3	210	0	0	3.406105	10.0473	167	

.

Figura 31 - Captura Estatística *Conversations* no cenário *multicast*.

Analisando detalhadamente a figura 28 conseguimos perceber que no cenário *unicast* apenas existiu uma conversação que ocupou uma largura de banda de 171 kb/s. Por outro lado, no cenário *multicast* (figura 30) apenas houve uma conversação que corresponde à *stream*, com largura de banda de 140 kb/s. Neste caso é de notar que a *stream* está a ser consumida simultaneamente por 4 clientes.

Observando o cenário *unicast*, percebemos que a maior vantagem a notar é a fácil configuração do mesmo, em que o transmissor possui um canal de transmissão para receber/responder aos pedidos de cada cliente, e cada cliente irá receber uma cópia dos dados de *streaming* enviados pelo servidor. Tendo isto em conta consegue-se perceber que um problema será o elevado tráfego na rede. Quantos mais pedidos o servidor receber, maior largura de banda necessitará para atender aos mesmos. Tendo em conta que a largura de banda é um recurso limitado, estamos perante um problema de escalabilidade, que poderá afetar a qualidade do serviço no lado do cliente.

Quanto ao *multicast*, o servidor envia uma cópia dos dados de *streaming* que será compartilhada por todos os clientes na rede interessados (transmissão feita para vários clientes em simultâneo). Em comparação ao cenário *unicast*, o serviço é mais escalável, visto que o aumento do número de clientes não afetará de forma tão abrupta a qualidade de transmissão. O tráfego na rede será também muito mais reduzido. Assim concluímos que este cenário ajuda a economizar banda e reduz significativamente a carga na rede. Este cenário possui algumas desvantagens como não garantir que os pacotes sejam entregues, nem que cheguem ao destino de forma ordenada. Outra desvantagem é que mesmo que não haja clientes ele está sempre a transmitir, ou seja a gastar banda desnecessariamente. Outro aspecto a notar é que os ISP não encaminham pacotes *multicast*.

Assim, a escolha entre *unicast* e *multicast* depende bastante do cenário e da rede em questão.

Em suma*, unicast* é mais simples de implementar, mas menos eficiente em termos de tráfego de rede quando o número de clientes é alto e *multicast* é escalável e eficiente (tráfego de rede), quando está bem configurado, mas é mais complexo de implementar.

Conclusão

Ao experimentar as várias soluções de *streaming* a pedido e em tempo real emuladas no core, o grupo constatou que cada cenário de *streaming* pode ser útil e apropriado dependendo de necessidades específicas e das condições da rede e recursos disponíveis.

Quanto ao *streaming* HTTP simples sem adaptação dinâmica de débito identificamos que é simples de implementar, tem grande compatibilidade e é fácil de escalar, porém falha na adaptação dinâmica de taxa de bits, o que pode levar a interrupções na reprodução em conexões de qualidade variável e é ineficiente em termos de largura de banda, uma vez que não ajusta a qualidade do vídeo de acordo com a largura de banda disponível. É por isso aconselhável em casos de *streaming* de conteúdo de baixa complexidade que não requer adaptação de qualidade, casos em que a simplicidade e a compatibilidade são mais importantes do que a otimização de largura de banda.

Quanto ao *streaming* adaptativo sobre HTTP (MPEG-DASH) identificamos que permite a adaptação dinâmica de taxa de bits, permitindo uma reprodução suave em diferentes condições de rede e melhora a eficiência de largura de banda, porém requer maior complexidade de implementação em comparação com o *streaming* HTTP simples. É por isso aconselhável a *streaming* de conteúdo de alta qualidade em ambientes com largura de banda variável, uma vez que, a qualidade de vídeo adapta-se dinamicamente à largura de banda disponível.

Quanto ao *streaming* RTP/RTCP *unicast* sobre UDP e *multicast* com anúncios SAP identificamos que o *multicast* é mais eficiente em termos de largura de banda, uma vez que reduz a replicação de dados na rede enquanto o *unicast* e ainda é escalável para um grande número de receptores, por outro lado, aloca largura de banda separada para cada receptor e tende a ter latência mais baixa, uma vez que os dados são entregues diretamente a cada receptor. Porém *multicast* necessita de uma infraestrutura de rede que o suporte e a sua configuração e manutenção são mais complexas, e quanto ao *unicast* é bastante ineficiente em termos de largura de banda pois sobrecarrega a mesma com dados replicados uma vez que cada receptor recebe uma das réplicas. É por isso adequado a *streaming* de eventos ao vivo, em redes com suporte a *multicast* e onde a eficiência de largura de banda é crítica.