

Engenharia de Serviços em Rede - TP1 - PL26

Afonso Xavier Cardoso Marques :: PG53601
Almerindo Valdemar Tchivangulula :: ID9861
Sérgio Ribeiro :: PG54708

Outubro 2023



Universidade do Minho
Escola de Engenharia

1 Introdução

O presente relatório foi desenvolvido no âmbito da unidade curricular de Engenharia de Serviços em Rede como proposta de resolução ao Trabalho Prático 1. Os leitores encontram no documento as conclusões que o grupo retirou face aos resultados empíricos do trabalho experimental proposto pela equipa docente.

Contents

1	Introdução	2
2	Parte Experimental - Questões e Respostas	4
2.1	Etapa 1 - Streaming HTTP simples sem adaptação dinâmica de débito	4
2.2	Etapa 2 - Streaming adaptativo sobre HTTP (MPEG-DASH) . .	13
2.3	Etapa 3 - Streaming RTP/RTCP unicast sobre UDP e multicast com anúncios SAP	18
3	Comentários Finais	20
4	Anexos	21
4.1	Etapa 1 - Resultados dos streams nos hosts Jasmine, Bela e Monstro	21
4.2	Etapa 2 - Resultados dos streams nos hosts Bela e Alladin . . .	22

2 Parte Experimental - Questões e Respostas

Para o trabalho experimental, a equipa docente disponibilizou uma topologia de rede que se baseia em cinco hosts, dois switches e dois routers.

Os resultados são levados a cabo pela interação entre os 5 hosts: **VStreamer**, **Jasmine**, **Alladin**, **Bela** e **Monstro**.

A baixo colocamos uma figura com a topologia e os destaques supramencionados.

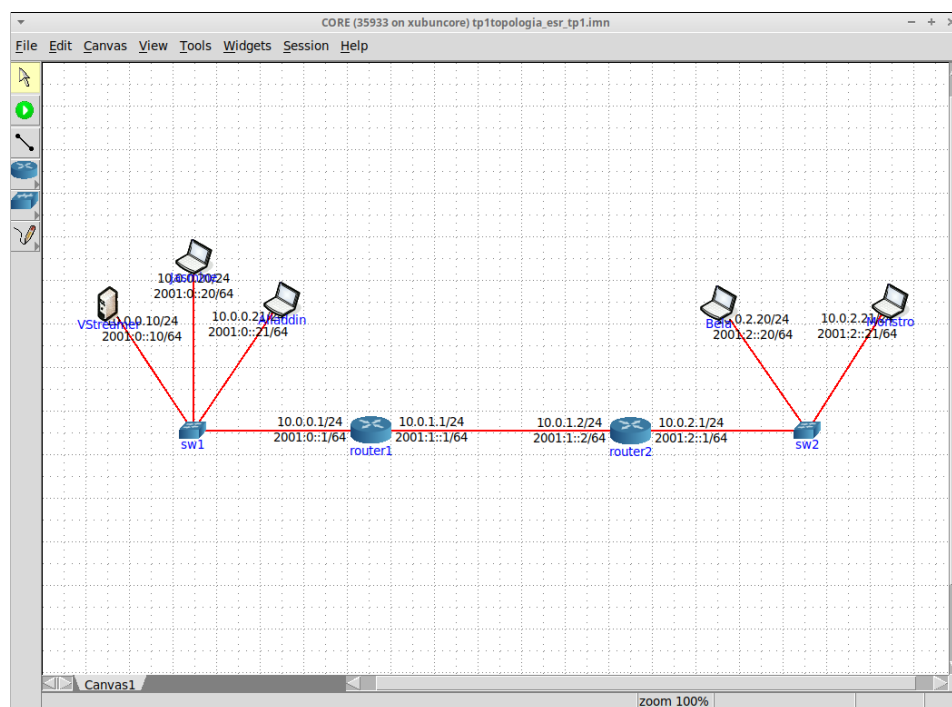


Figure 1: Topologia e os seus destaques

2.1 Etapa 1 - Streaming HTTP simples sem adaptação dinâmica de débito

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

A taxa em bps esperada pode ser consultada através do comando `ffmpeg -i videoA.mp4`, como podemos ver na figura abaixo.



Nas figuras 4 e 5 podemos ver que a taxa para tráfego com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffmpeg) é 18kbps e 32kbps respectivamente.

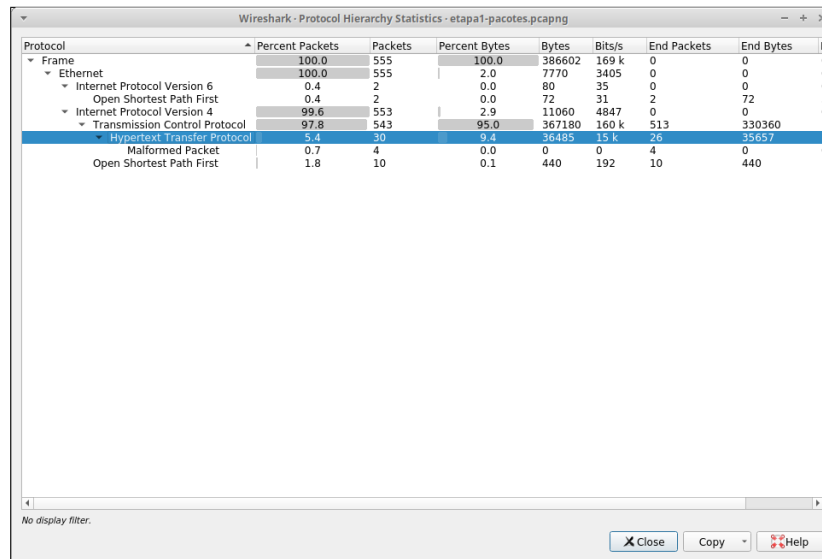


Figure 3: Protocol Hierarchy do tráfego com VStreamer e Jasmine

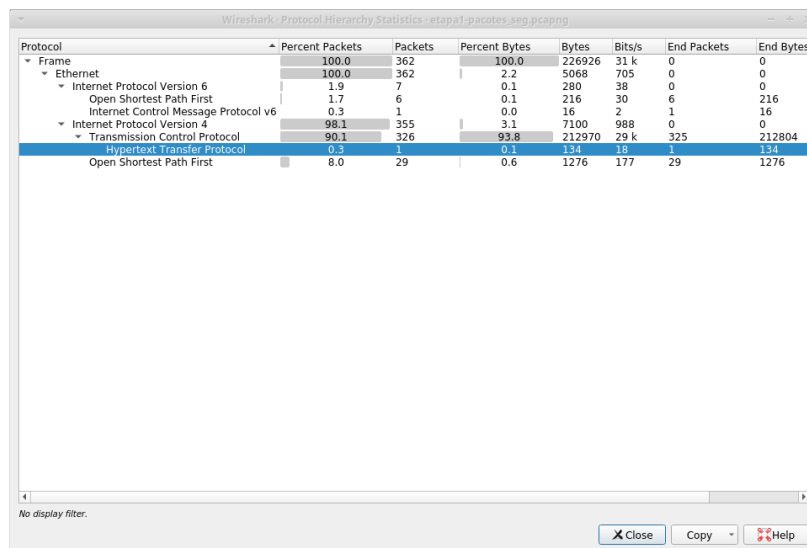


Figure 4: Protocol Hierarchy do tráfego com VStreamer e Jasmine + Bela

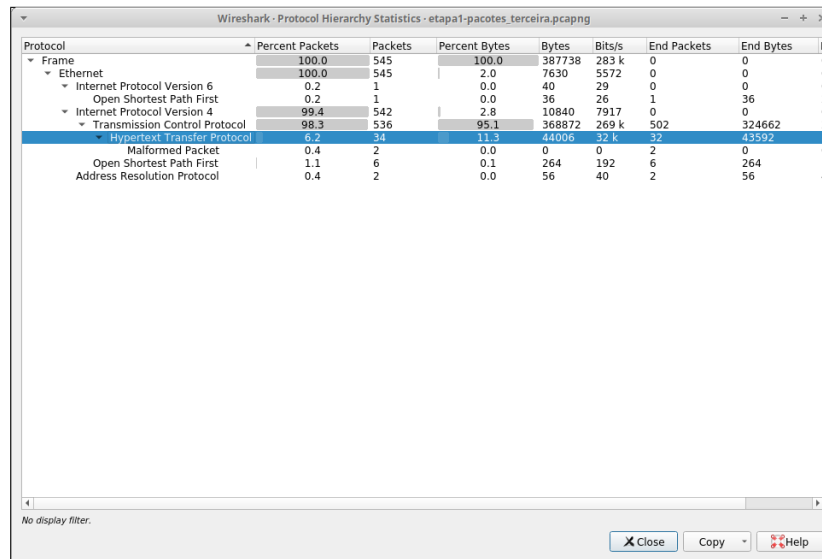


Figure 5: Protocol Hierarchy do tráfego Com VStreamer e Jasmine + Bela + Monstro

Pelas figuras 6, 7 e 8 podemos ver as capturas das tramas TCP do Wireshark que mostram que houve encapsulamento dos pacotes transmitidos nos diferentes níveis da pilha protocolar, ou seja, na camada de transporte (TCP), na de rede (IPv4), na de aplicação (HTTP) e na de dados (Ethernet).

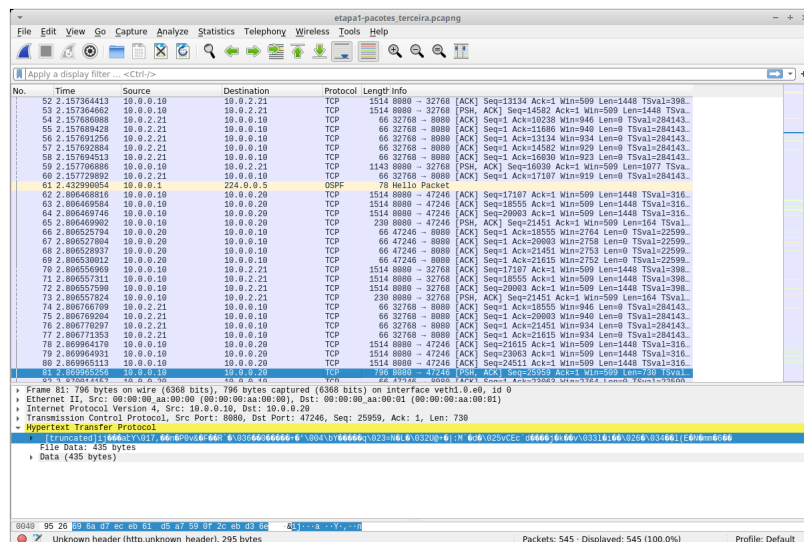
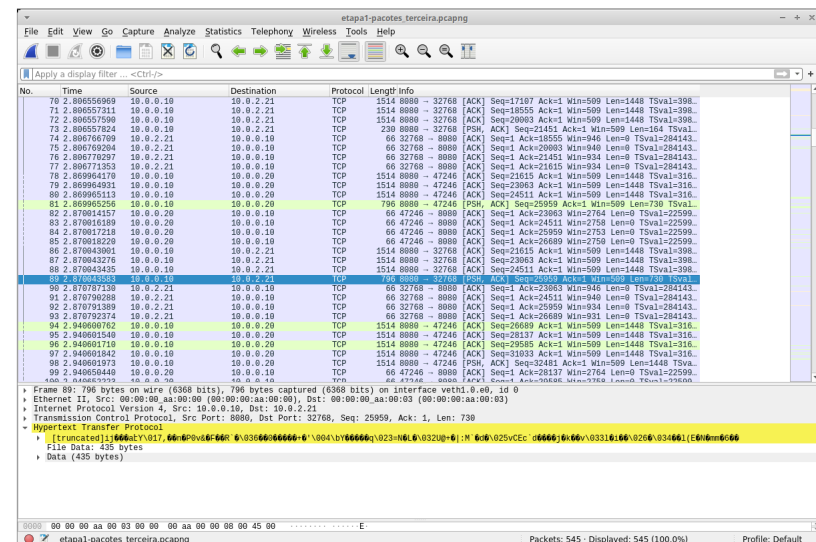
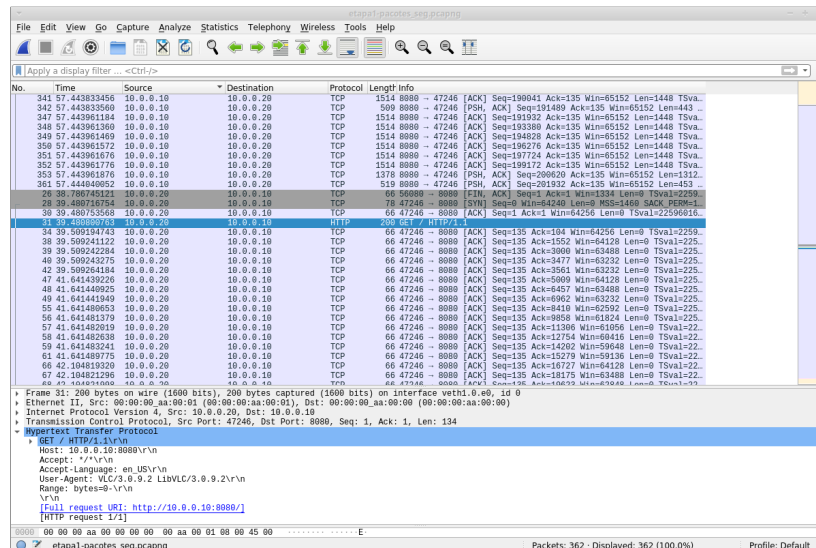
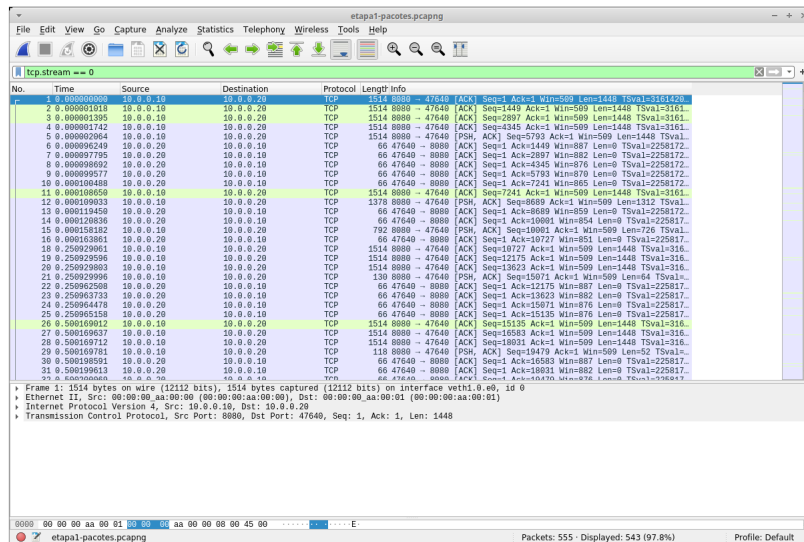


Figure 6: Tráfego com servidor e Jasmine (1 cliente)

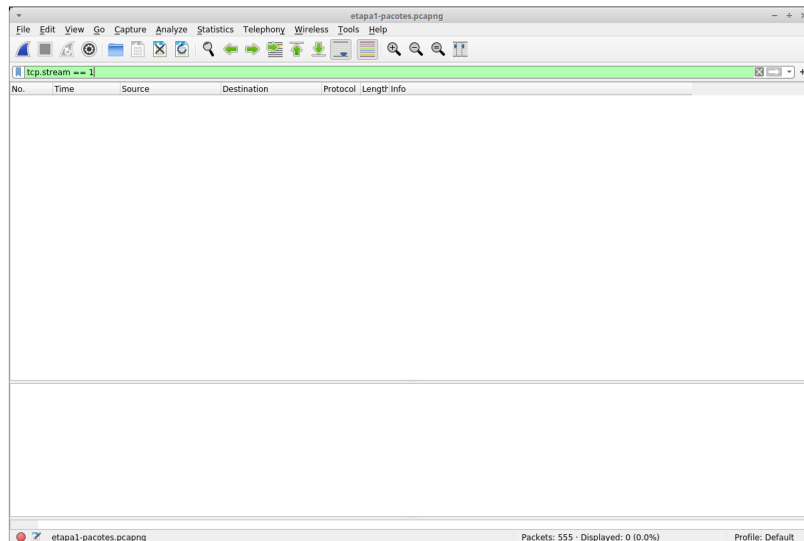


Relativamente ao número total de fluxos gerados recorremos ao filtro tcp.stream. Para o caso de apenas 1 cliente VLC pudemos observar que ocorreu 1 fluxo, visto que apenas existe a stream 0. Isto corresponde ao cliente transmitir no VLC.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=1 Acks=1 Win=509 Len=1448 TSval=3161426
2	0.000001918	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=1449 Acks=1 Win=509 Len=1448 TSval=31615
3	0.000001395	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=2097 Acks=1 Win=509 Len=1448 TSval=31615
4	0.000001742	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=4345 Acks=1 Win=509 Len=1448 TSval=31615
5	0.000002054	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [PSH, ACK] Seq=5703 Acks=1 Win=509 Len=1448 TSval=31615
6	0.000002249	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=1449 Win=887 Len=0 TSval=2258172
7	0.000007795	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=2087 Win=882 Len=0 TSval=2258172
8	0.000006692	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=4345 Win=876 Len=0 TSval=2258172
9	0.000009577	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=5703 Win=870 Len=0 TSval=2258172
10	0.000100488	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=7241 Win=885 Len=0 TSval=2258172
11	0.000108050	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=7241 Acks=1 Win=509 Len=1448 TSval=31615
12	0.000109033	10.0.0.10	10.0.0.20	TCP	1370	8880 → 47640 [PSH, ACK] Seq=8689 Acks=1 Win=509 Len=1312 TSval=31615
13	0.000115450	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=8689 Win=859 Len=0 TSval=2258172
14	0.000128836	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=10001 Win=854 Len=0 TSval=2258172
15	0.000158182	10.0.0.10	10.0.0.20	TCP	782	8880 → 47640 [PSH, ACK] Seq=10001 Acks=1 Win=509 Len=728 TSval=31615
16	0.000163861	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=10727 Win=851 Len=0 TSval=2258172
17	0.250925961	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=10727 Acks=1 Win=509 Len=1448 TSval=31615
18	0.250925986	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=12175 Acks=1 Win=509 Len=1448 TSval=31615
19	0.250925993	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=12623 Acks=1 Win=509 Len=1448 TSval=31615
20	0.250925996	10.0.0.10	10.0.0.20	TCP	130	8880 → 47640 [PSH, ACK] Seq=15071 Acks=1 Win=509 Len=64 TSval=31615
21	0.250925999	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=12175 Win=887 Len=0 TSval=2258172
22	0.250962508	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=13023 Win=882 Len=0 TSval=2258172
23	0.250963733	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=15071 Win=876 Len=0 TSval=2258172
24	0.250964478	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=15135 Win=876 Len=0 TSval=2258172
25	0.250965158	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=15135 Win=876 Len=0 TSval=2258172
26	0.500109012	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=15135 Acks=1 Win=509 Len=1448 TSval=31615
27	0.500109037	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=16583 Acks=1 Win=509 Len=1448 TSval=31615
28	0.500109712	10.0.0.10	10.0.0.20	TCP	1514	8880 → 47640 [ACK] Seq=18031 Acks=1 Win=509 Len=1448 TSval=31615
29	0.500109781	10.0.0.10	10.0.0.20	TCP	118	8880 → 47640 [PSH, ACK] Seq=19479 Acks=1 Win=509 Len=52 TSval=31615
30	0.500109521	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=16583 Win=887 Len=0 TSval=2258172
31	0.500109612	10.0.0.10	10.0.0.10	TCP	66	47640 → 8880 [ACK] Seq=1 Acks=18031 Win=882 Len=0 TSval=2258172

Figure 9: Stream 0 para 1 cliente



No.	Time	Source	Destination	Protocol	Length	Info
-----	------	--------	-------------	----------	--------	------

Figure 10: Stream 1 para 1 cliente

Com 2 clientes (VLC e Firefox) foram gerados 2 fluxos, uma vez que existem as streams 0 e 1. Estas correspondem aos serviços VLC e Firefox.

No.	Time	Source	Destination	Protocol	Length	Info
26	39.786745121	10.0.0.20	10.0.0.10	TCP	66	56880 → 8080 [FIN, ACK] Seq=1 Ack=1 Win=1334 Len=0 TSval=2259
27	39.786767783	10.0.0.10	10.0.0.20	TCP	66	8080 → 56880 [ACK] Seq=1 Ack=2 Win=509 Len=0 TSval=3162849326

Figure 11: Stream 0 para 2 clientes

No.	Time	Source	Destination	Protocol	Length	Info
28	39.489716154	10.0.0.20	10.0.0.10	TCP	78	47240 → 8080 [SYN, ACK] Seq=0 Win=64240 Len=0 MSS=1460 SACK Per=1
30	39.489753568	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=22596016
31	39.489809763	10.0.0.20	10.0.0.10	HTTP	200	GET / HTTP/1.1
32	39.489804817	10.0.0.10	10.0.0.20	TCP	66	8080 → 47240 [ACK] Seq=1 Ack=135 Win=65152 Len=0 TSval=316285
33	39.509158429	10.0.0.10	10.0.0.20	TCP	109	8080 → 47240 [PSH, ACK] Seq=1 Ack=135 Win=65152 Len=103 TSval=
34	39.509184743	10.0.0.10	10.0.0.20	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=184 Win=64256 Len=0 TSval=2259
35	39.509223514	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=184 Ack=135 Win=65152 Len=1448 TSval=3
36	39.509223722	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=1552 Ack=135 Win=65152 Len=1448 TSval=
37	39.509223878	10.0.0.10	10.0.0.20	TCP	543	8080 → 47240 [PSH, ACK] Seq=3008 Ack=135 Win=65152 Len=477 TS
38	39.509241122	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=1552 Win=64128 Len=0 TSval=225
39	39.509242284	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=3008 Win=65488 Len=0 TSval=225
40	39.509243275	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=3477 Win=63232 Len=0 TSval=225
41	39.509255951	10.0.0.10	10.0.0.20	TCP	159	8080 → 47240 [PSH, ACK] Seq=3477 Ack=135 Win=65152 Len=84 TSv
42	39.509264184	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=3561 Win=63232 Len=0 TSval=225
44	41.641398312	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=5951 Ack=135 Win=65152 Len=1448 TSval=
45	41.641399006	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=5909 Ack=135 Win=65152 Len=1448 TSval=
46	41.641399191	10.0.0.10	10.0.0.20	TCP	571	8080 → 47240 [PSH, ACK] Seq=6457 Ack=135 Win=65152 Len=505 TS
47	41.641432026	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=5909 Win=64128 Len=0 TSval=225
48	41.641440925	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=6457 Win=63488 Len=0 TSval=225
49	41.641441949	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=6962 Win=63232 Len=0 TSval=225
50	41.641467017	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=6962 Ack=135 Win=65152 Len=1448 TSval=
51	41.641467184	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=8418 Ack=135 Win=65152 Len=1448 TSval=
52	41.641467387	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=8958 Ack=135 Win=65152 Len=1448 TSval=
53	41.641467498	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [ACK] Seq=11396 Ack=135 Win=65152 Len=1448 TSval=
54	41.641467587	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47240 [PSH, ACK] Seq=12754 Ack=135 Win=65152 Len=1448
55	41.641480653	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=8418 Win=62592 Len=0 TSval=225
56	41.641481379	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=8958 Win=61824 Len=0 TSval=225
57	41.641482919	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=11396 Win=61824 Len=0 TSval=22
58	41.641482638	10.0.0.20	10.0.0.10	TCP	66	47240 → 8080 [ACK] Seq=135 Ack=12754 Win=68416 Len=0 TSval=22

Figure 12: Stream 1 para 2 clientes

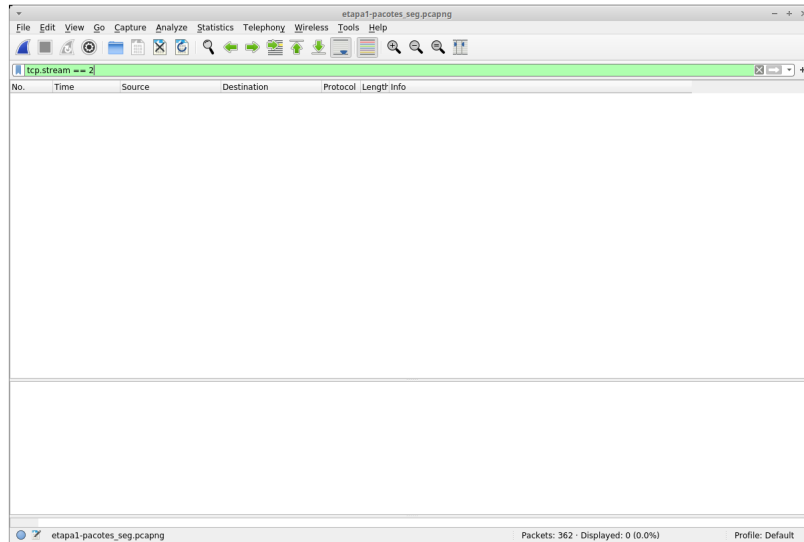


Figure 13: Stream 2 para 2 clientes

Para 3 clientes (VLC, Firefox e fflay) foram gerados 3 fluxos com as streams 0, 1 e 2, correspondentes aos serviços VLC, Firefox e fflay.

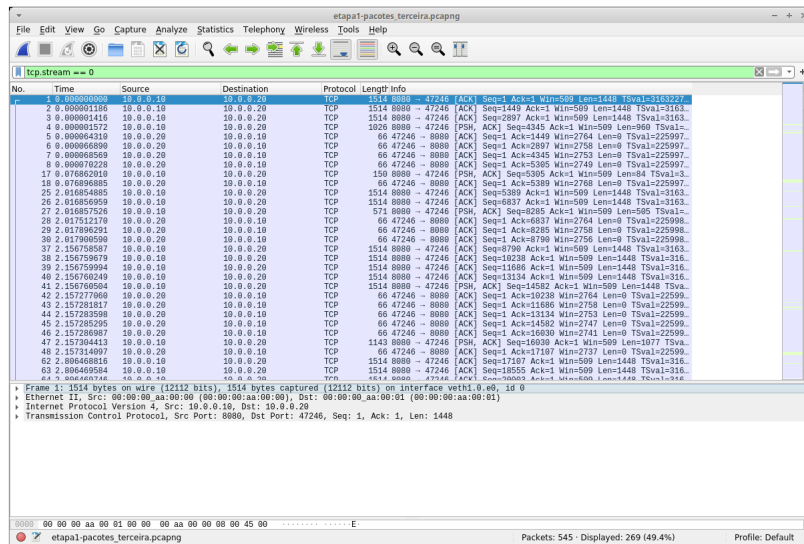


Figure 14: Stream 0 para 3 clientes

No.	Time	Source	Destination	Protocol	Length	Info
9	0.00011945	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=1 Acks=1 Win=509 Len=1448 Tsv=1398220
10	0.00021217	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=1449 Acks=1 Win=509 Len=1448 Tsv=13986
11	0.000212548	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=2097 Acks=1 Win=509 Len=1448 Tsv=13986
12	0.000212901	10.0.0.10	10.0.0.21	TCP	1026	8080 → 32768 [PSH, ACK] Seq=4345 Acks=1 Win=509 Len=800 Tsv=13986
13	0.000404903	10.0.0.10	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=1440 Win=940 Len=0 Tsv=12841429
14	0.000853166	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=2097 Win=940 Len=0 Tsv=12841429
15	0.000854209	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=4345 Win=934 Len=0 Tsv=12841429
16	0.000855381	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=5385 Win=931 Len=0 Tsv=12841429
19	0.076918485	10.0.0.10	10.0.0.21	TCP	150	8080 → 32768 [PSH, ACK] Seq=5385 Acks=1 Win=509 Len=84 Tsv=13986
20	0.076925313	10.0.0.10	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=5389 Win=950 Len=0 Tsv=12841429
31	2.026398887	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=5389 Acks=1 Win=509 Len=1448 Tsv=13986
32	2.026399289	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=8037 Acks=1 Win=509 Len=1448 Tsv=13986
33	2.026399491	10.0.0.10	10.0.0.21	TCP	571	8080 → 32768 [PSH, ACK] Seq=8285 Acks=1 Win=509 Len=585 Tsv=13986
34	2.027888461	10.0.0.10	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=8037 Win=940 Len=0 Tsv=12841431
35	2.027817757	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=8285 Win=940 Len=0 Tsv=12841431
36	2.027826517	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=8799 Win=938 Len=0 Tsv=12841431
40	2.157363488	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=8799 Acks=1 Win=509 Len=1448 Tsv=13986
50	2.157363897	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=10238 Acks=1 Win=509 Len=1448 Tsv=13986
51	2.157364172	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=11686 Acks=1 Win=509 Len=1448 Tsv=13986
52	2.157364413	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=13134 Acks=1 Win=509 Len=1448 Tsv=13986
53	2.157364662	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [PSH, ACK] Seq=14582 Acks=1 Win=509 Len=1448 Tsv=13986
54	2.157368688	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=10238 Win=940 Len=0 Tsv=12841431
55	2.1573689428	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=11686 Win=940 Len=0 Tsv=12841431
56	2.157369256	10.0.0.10	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=13134 Win=934 Len=0 Tsv=12841431
57	2.1573692884	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=14582 Win=929 Len=0 Tsv=12841431
58	2.1573694513	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=16030 Win=921 Len=0 Tsv=12841431
59	2.157706886	10.0.0.10	10.0.0.21	TCP	1463	8080 → 32768 [PSH, ACK] Seq=16030 Acks=1 Win=509 Len=1077 Tsv=13986
60	2.157728892	10.0.0.21	10.0.0.10	TCP	66	32768 → 8080 [ACK] Seq=1 Acks=17187 Win=910 Len=0 Tsv=12841431
70	2.880555969	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=17187 Acks=1 Win=509 Len=1448 Tsv=13986
71	2.880557311	10.0.0.10	10.0.0.21	TCP	1514	8080 → 32768 [ACK] Seq=18555 Acks=1 Win=509 Len=1448 Tsv=13986

Frame 9: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth1.0.0.0, Id 0
 Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:03 (00:00:00:aa:00:03)
 Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.21
 Transmission Control Protocol, Src Port: 8080, Dst Port: 32768, Seq: 1, Ack: 1, Len: 1448

Packets: 545 · Displayed: 267 (49.0%) · Profile: Default

Figure 15: Stream 1 para 3 clientes

No.	Time	Source	Destination	Protocol	Length	Info
316	5.924703983	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=53718 Acks=1 Win=509 Len=1448 Tsv=11751934
317	5.924704048	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=65166 Acks=1 Win=509 Len=1448 Tsv=11751934
318	5.924704121	10.0.0.10	10.0.0.20	TCP	180	8080 → 44440 [PSH, ACK] Seq=66014 Acks=1 Win=509 Len=120 Tsv=11751934
319	5.924723857	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=63718 Win=509 Len=0 Tsv=1380855441
320	5.924724267	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=65166 Win=509 Len=0 Tsv=1380855441
321	5.924724775	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=66014 Win=509 Len=0 Tsv=1380855441
322	5.924725184	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=66734 Win=509 Len=0 Tsv=1380855441
339	6.423751074	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=66734 Acks=1 Win=509 Len=1448 Tsv=11751934
340	6.423751159	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=68182 Acks=1 Win=509 Len=1448 Tsv=11751934
341	6.423751223	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=69630 Acks=1 Win=509 Len=1448 Tsv=11751934
342	6.423751297	10.0.0.10	10.0.0.20	TCP	418	8080 → 44440 [PSH, ACK] Seq=71078 Acks=1 Win=509 Len=352 Tsv=11751934
343	6.423779806	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=68182 Win=509 Len=0 Tsv=1380855440
344	6.423771299	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=69630 Win=509 Len=0 Tsv=1380855440
345	6.423771708	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=71078 Win=509 Len=0 Tsv=1380855440
346	6.423772126	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=71430 Win=509 Len=0 Tsv=1380855440
364	6.912456284	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=71430 Acks=1 Win=509 Len=1448 Tsv=11751944
365	6.912456387	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=72878 Acks=1 Win=509 Len=1448 Tsv=11751944
366	6.912456403	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=74326 Acks=1 Win=509 Len=1448 Tsv=11751944
367	6.912456526	10.0.0.10	10.0.0.20	TCP	75	8080 → 44440 [PSH, ACK] Seq=75774 Acks=1 Win=509 Len=0 Tsv=11751944
368	6.912477106	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=72878 Win=509 Len=0 Tsv=1380855428
369	6.912477692	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=74326 Win=509 Len=0 Tsv=1380855428
370	6.912478011	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=75774 Win=509 Len=0 Tsv=1380855428
371	6.912478414	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=75783 Win=509 Len=0 Tsv=1380855428
388	7.416179798	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=75783 Acks=1 Win=509 Len=1448 Tsv=11751948
389	7.416179898	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=77231 Acks=1 Win=509 Len=1448 Tsv=11751948
390	7.416179965	10.0.0.10	10.0.0.20	TCP	1514	8080 → 44440 [ACK] Seq=78670 Acks=1 Win=509 Len=1448 Tsv=11751948
391	7.416180039	10.0.0.10	10.0.0.20	TCP	66	8080 → 44440 [PSH, ACK] Seq=80127 Acks=1 Win=509 Len=0 Tsv=11751948
392	7.416200008	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=77231 Win=509 Len=0 Tsv=1380855532
393	7.416200076	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=78670 Win=509 Len=0 Tsv=1380855532
394	7.416200089	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=80127 Win=509 Len=0 Tsv=1380855532
395	7.416200123	10.0.0.20	10.0.0.10	TCP	66	44440 → 8080 [ACK] Seq=1 Acks=80127 Win=509 Len=0 Tsv=1380855532

Frame 395: 60 bytes on wire (528 bits), 60 bytes captured (528 bits) on interface veth1.0.0.1, 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: 10.0.0.20, Dst: 10.0.0.10
 Transmission Control Protocol, Src Port: 44440, Dst Port: 8080, Seq: 1, Ack: 80129, Len: 0

Figure 16: Stream 2 para 3 clientes

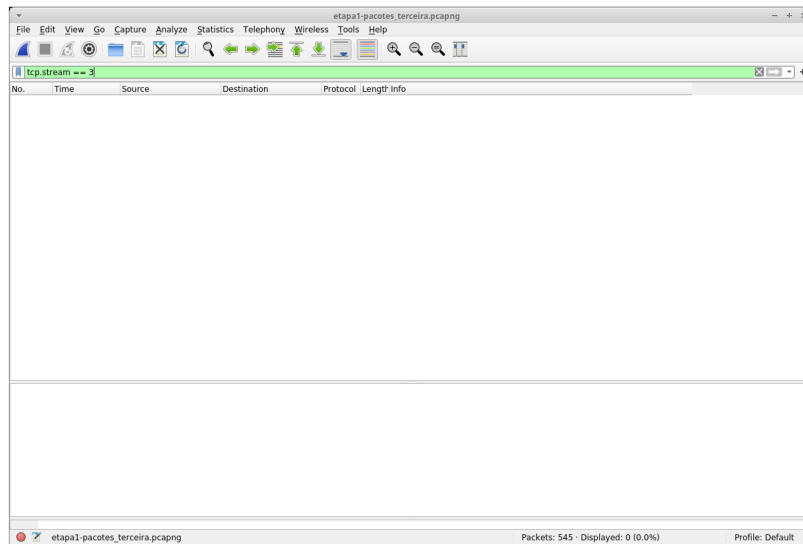


Figure 17: Stream 3 para 3 clientes

Finalmente, a solução não terá uma boa escalabilidade dado que as respostas aos pedidos são enviadas individualmente, o que implicaria que se fossem feitos pedidos iguais as tramas seriam enviadas para todos os clientes na mesma.

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

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 de banda pode ser consultada no ficheiro "*video_manifest.mpd*".

Como podemos ver pelas três figuras seguintes, a largura de banda necessária para visualizar o vídeo *videoB_200_150_200k.mp4* é 151057 bps. Por sua vez, para ver o *videoB_480_360_500k.mp4* é precisa uma taxa de 415928 bps. E por fim, o *videoB_640_480_1000k.mp4* necessita de uma largura de banda de 772531 bps.

```
Terminal - core@xubuncore:~  
File Edit View Terminal Tabs Help  
[?]xml version="1.0"?>  
[?]MPD file Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gc5ad4e4+dfsg5-5 at 2023-10-04T09:12:33.776Z->  
[?]MPD xmlns:urn:mpeg:dash:schema:mpd:2011="http://dash.mp4/" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H0M5.230S" maxSegmentDuration="PT0H0M0.502S" profile="urn:mpeg:dash:profile:full2011" />  
[?]<!-- Program Information -->  
[?]<ProgramInformation moreInformationURL="http://gpac.sourceforge.net">  
[?]<Title>video_manifest.mpd generated by GPAC</Title>  
[?]</ProgramInformation>  
[?]<Period duration="PT0H0M5.230S">  
[?]<AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="640" maxHeight="480" maxFrameRate="11456/384" par="4:3" lang="und">  
[?]<SegmentList>  
[?]<Initialization sourceURL="video_manifest_init.mpd"/>  
[?]</SegmentList>  
[?]<Representation id="1" mimeType="video/mp4" codecs="avc3.64000e" width="280" height="150" frameRate="11456/384" sar="1:1" startWithSAP="0" bandwidth="151057">  
[?]<BaseURL>videoB_280_150_280k.dash.mpd</BaseURL>  
[?]<SegmentList timescale="11456" duration="5750">  
[?]<SegmentURL mediaRange="927-7881" indexRange="927-978"/>  
[?]<SegmentURL mediaRange="7882-13199" indexRange="7882-7125"/>  
[?]<SegmentURL mediaRange="13200-19079" indexRange="13200-13243"/>  
[?]<SegmentURL mediaRange="19080-21511" indexRange="19080-19123"/>  
[?]<SegmentURL mediaRange="31512-37325" indexRange="31512-31555"/>  
[?]<SegmentURL mediaRange="37326-45980" indexRange="37326-37369"/>  
[?]<SegmentURL mediaRange="45981-62352" indexRange="45981-46024"/>  
[?]<SegmentURL mediaRange="62353-71073" indexRange="62353-62396"/>  
[?]<SegmentURL mediaRange="71074-80412" indexRange="71074-71117"/>  
[?]<SegmentURL mediaRange="80413-87080" indexRange="80413-80456"/>  
[?]<SegmentURL mediaRange="87081-98735" indexRange="87081-87124"/>  
[?]</SegmentList>  
[?]</Representation>
```

Figure 18: Largura de banda para o *videoB_200_150_200k.mpd*

```
Terminal - core@xubuncore:~  
File Edit View Terminal Tabs Help  
[?]<Representation id="2" mimeType="video/mp4" codecs="avc3.64001e" width="480" height="360" frameRate="11456/384" sar="1:1" startWithSAP="0" bandwidth="415928">  
[?]<BaseURL>videoB_480_360_500k.dash.mpd</BaseURL>  
[?]<SegmentList timescale="11456" duration="5750">  
[?]<SegmentURL mediaRange="927-15163" indexRange="927-978"/>  
[?]<SegmentURL mediaRange="15164-36342" indexRange="15164-15207"/>  
[?]<SegmentURL mediaRange="36343-53137" indexRange="36343-36386"/>  
[?]<SegmentURL mediaRange="53138-90248" indexRange="53138-53181"/>  
[?]<SegmentURL mediaRange="90249-104363" indexRange="90249-90292"/>  
[?]<SegmentURL mediaRange="104364-126922" indexRange="104364-104407"/>  
[?]<SegmentURL mediaRange="126923-173720" indexRange="126923-126976"/>  
[?]<SegmentURL mediaRange="173721-197709" indexRange="173721-173764"/>  
[?]<SegmentURL mediaRange="197710-221812" indexRange="197710-197753"/>  
[?]<SegmentURL mediaRange="221813-239570" indexRange="221813-221856"/>  
[?]<SegmentURL mediaRange="239570-271863" indexRange="239570-239613"/>  
[?]</SegmentList>  
[?]</Representation>
```

Figure 19: Largura de banda para o *videoB_480_360_500k.mpd*

```
Terminal - core@xubuncore:~  
File Edit View Terminal Tabs Help  
[?]</Representation>  
[?]<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="480" frameRate="11456/384" sar="1:1" startWithSAP="0" bandwidth="772531">  
[?]<BaseURL>videoB_640_480_1000k.dash.mpd</BaseURL>  
[?]<SegmentList timescale="11456" duration="5750">  
[?]<SegmentURL mediaRange="927-27199" indexRange="927-978"/>  
[?]<SegmentURL mediaRange="27200-62840" indexRange="27200-27243"/>  
[?]<SegmentURL mediaRange="62841-89167" indexRange="62841-62884"/>  
[?]<SegmentURL mediaRange="89168-163374" indexRange="89168-89211"/>  
[?]<SegmentURL mediaRange="163375-188592" indexRange="163375-163418"/>  
[?]<SegmentURL mediaRange="188593-232464" indexRange="188593-188636"/>  
[?]<SegmentURL mediaRange="232465-323077" indexRange="232465-232508"/>  
[?]<SegmentURL mediaRange="323078-357870" indexRange="323078-323121"/>  
[?]<SegmentURL mediaRange="357880-408535" indexRange="357880-357923"/>  
[?]<SegmentURL mediaRange="408536-441470" indexRange="408536-408579"/>  
[?]<SegmentURL mediaRange="441471-504950" indexRange="441471-441514"/>  
[?]</SegmentList>
```

Figure 20: Largura de banda para o *videoB_640_480_1000k.mpd*

Na figura seguinte conseguimos ver pacotes com o protocolo HTTP na camada de aplicação que foram captados durante os streams dos vídeos em Bela e Alladin com o VStreamer. Destacamos dois pacotes em particular onde conseguimos ver os pedidos (GET) feitos por Alladin (10.0.0.21) e por Bela (10.0.2.20) ao VStreamer (10.0.0.10) a requisitar os dados do vídeo.

No.	Time	Source	Destination	Protocol	Length	Info
13550	10.75923902	10.0.2.20	10.0.0.10	HTTP	466	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
14172	10.764353149	10.0.0.10	10.0.2.20	MP4	1318	
14185	10.815215929	10.0.2.20	10.0.0.10	HTTP	466	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
14791	10.820035513	10.0.0.10	10.0.2.20	MP4	1318	
14806	23.997183112	10.0.0.21	10.0.0.10	HTTP	477	GET /video_dash.html HTTP/1.1
14808	23.997677896	10.0.0.10	10.0.0.21	HTTP	272	HTTP/1.1 304 Not Modified
14815	24.405481916	10.0.0.21	10.0.0.10	HTTP	381	GET /favicon.ico HTTP/1.1
14817	24.406100236	10.0.0.10	10.0.0.21	HTTP	741	HTTP/1.1 404 Not Found (text/html)
14824	24.418163358	10.0.0.21	10.0.0.10	HTTP	505	GET /video_manifest.mpd HTTP/1.1
14826	24.418748868	10.0.0.10	10.0.0.21	HTTP	273	HTTP/1.1 304 Not Modified
14832	25.652272728	10.0.0.21	10.0.0.10	HTTP	421	GET /video_manifest_init.mp4 HTTP/1.1
14834	25.653385759	10.0.0.10	10.0.0.21	HTTP	257	HTTP/1.1 304 Not Modified
14840	25.994311387	10.0.0.21	10.0.0.10	HTTP	411	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
15351	25.997890271	10.0.0.10	10.0.0.21	MP4	126	
15385	26.03184287	10.0.0.21	10.0.0.10	HTTP	413	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
15890	26.03889347	10.0.0.10	10.0.0.21	MP4	1318	
15914	26.039393535	10.0.0.21	10.0.0.10	HTTP	413	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
16413	26.104106201	10.0.0.10	10.0.0.21	MP4	1318	
16436	26.127950842	10.0.0.21	10.0.0.10	HTTP	414	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
17042	26.131787831	10.0.0.21	10.0.0.10	MP4	1318	
17049	26.151145553	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
17529	26.155152078	10.0.0.10	10.0.0.21	MP4	1318	
17540	26.168046801	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
18127	26.171423251	10.0.0.10	10.0.0.21	MP4	1318	
18142	26.195426728	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
18712	26.207576134	10.0.0.10	10.0.0.21	MP4	1318	
18718	26.220797080	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
19094	26.231398407	10.0.0.10	10.0.0.21	MP4	1318	
19101	26.249065063	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
19891	26.251955545	10.0.0.10	10.0.0.21	MP4	1318	
19897	26.273946100	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
20163	26.276583264	10.0.0.10	10.0.0.21	MP4	1318	
20188	26.294713106	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
20669	26.297619769	10.0.0.10	10.0.0.21	MP4	214	

Figure 21: Pacotes capturados durante o stream

Para sabermos qual a pilha protocolar usada basta observar um dos pacotes HTTP enviados, onde estão presentes os protocolos Ethernet (dados), IPv4 (rede), TCP (transporte) e HTTP (aplicação).

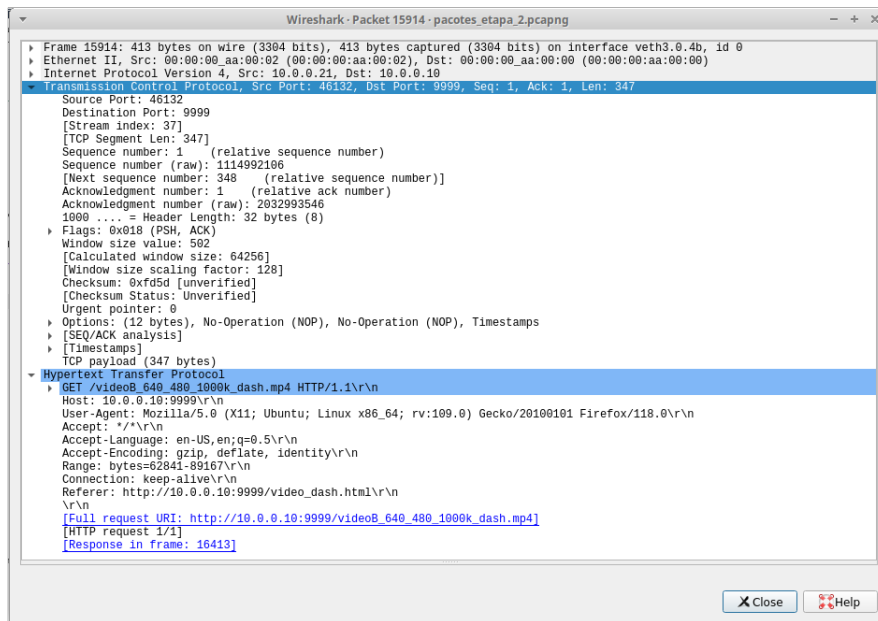


Figure 22: Análise da pilha protocolar de uma pacote HTTP

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.

De forma a restringir a largura de banda para o cliente no portátil Bela restringimos o link entre os dois routers de modo a que apenas fosse permitida a transmissão do videoB_200_150_200k.mp4. Para isso baseámo-nos na informação disponível no ficheiro "video_manifest.mpd", já consultada na alínea anterior.

Uma vez que a largura de banda necessária para o video de menor qualidade é de 151057 bps e para o de intermédia é de 415928 bps, restringimos o link entre os dois routers para 200000 bps. Conseguimos ver pelos pacotes captados que o Bela faz pedidos GET a começar pelo vídeo de melhor qualidade, descendo para o de qualidade intermédia até que se estabiliza no de qualidade mais baixa.

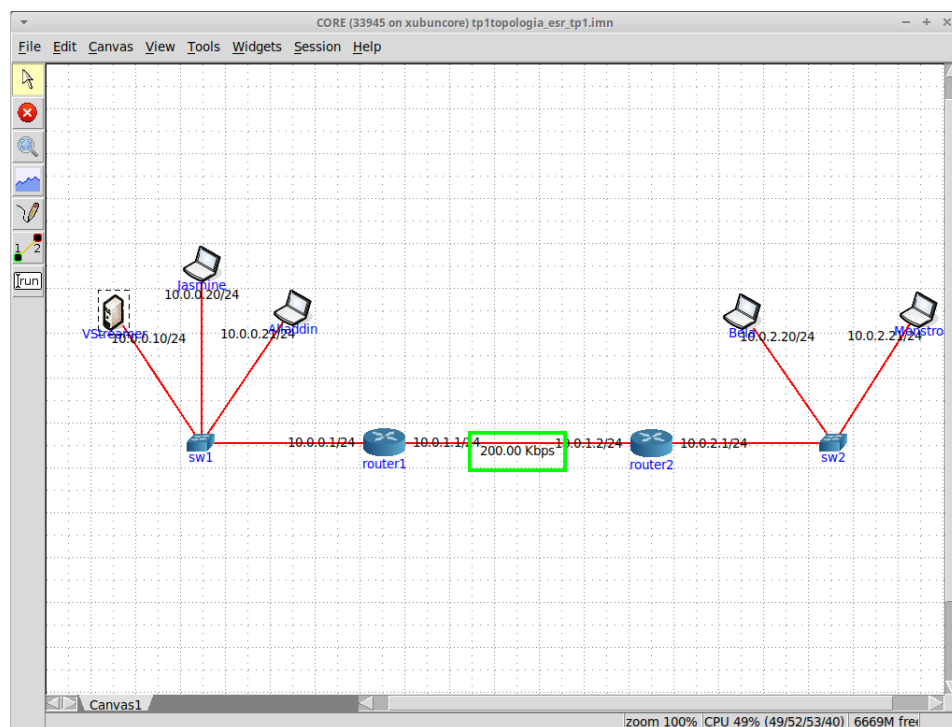


Figure 23: Topologia com limitação

8976	111	859897638	10.0.0.10	10.0.2.20	MP4	542	
9386	125	262058869	10.0.2.20	10.0.0.10	HTTP	411	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
9400	125	283123373	10.0.2.20	10.0.0.10	HTTP	412	GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
9416	125	309078379	10.0.2.20	10.0.0.10	HTTP	412	GET /videoB_480_360_500k_dash.mp4 HTTP/1.1
9451	125	453136785	10.0.2.20	10.0.0.10	HTTP	412	GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
9508	127	493385452	10.0.2.20	10.0.0.10	HTTP	412	GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
9619	129	966343790	10.0.0.10	10.0.2.20	MP4	542	

Figure 24: Pacotes captados relativos ao host Bela

Para que o cliente no portátil Alladin exiba o vídeo com mais resolução basta não limitar a ligação, ou seja, assumirmos que o limite seria infinito.

82	1.591599598	10.0.0.21	10.0.0.10	HTTP	477	GET /video_dash.html HTTP/1.1
84	1.592198544	10.0.0.10	10.0.0.21	HTTP	272	HTTP/1.1 304 Not Modified
108	2.008079261	10.0.0.21	10.0.0.10	HTTP	411	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
613	2.009556763	10.0.0.10	10.0.0.21	MP4	1318	
675	2.038742329	10.0.0.21	10.0.0.10	HTTP	413	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
1065	2.042299866	10.0.0.10	10.0.0.21	MP4	1318	
1074	2.100349388	10.0.0.21	10.0.0.10	HTTP	413	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
1582	2.194094163	10.0.0.10	10.0.0.21	MP4	1318	
1683	2.411559391	10.0.0.21	10.0.0.10	HTTP	414	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
2031	2.413163247	10.0.0.10	10.0.0.21	MP4	1318	
2056	2.431994127	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
2441	2.434643283	10.0.0.10	10.0.0.21	MP4	1318	
2448	2.452645231	10.0.0.21	10.0.0.10	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
3004	2.473110426	10.0.0.10	10.0.0.21	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
3499	2.475087654	10.0.0.21	10.0.0.10	MP4	1318	
3520	2.491379036	10.0.0.10	10.0.0.21	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
3969	2.493562787	10.0.0.21	10.0.0.10	MP4	1318	
4002	2.516756432	10.0.0.10	10.0.0.21	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
4560	2.519272845	10.0.0.21	10.0.0.10	MP4	1318	
4569	2.550452714	10.0.0.10	10.0.0.21	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
4955	2.552238114	10.0.0.21	10.0.0.10	MP4	1318	
4965	2.570178782	10.0.0.10	10.0.0.21	HTTP	415	GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
5416	2.573129381	10.0.0.21	10.0.0.10	MP4	1318	

Figure 25: Pacotes captados relativos ao host Alladin

O resultado final é o esperado, com o Bela a transmitir o video de menor qualidade (*videoB_200_150_200k.mp4*) e o Alladin a transmitir o de maior qualidade (*videoB_640_480_1000k.mp4*).

Questão 4: Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.

Traduzindo a definição retirada do livro da bibliografia recomendada pelos docentes - Computer Networking: A Top-Down Approach, 7th Edition - podemos descrever o DASH como:

”DASH permite aos clientes com diferentes taxas de acesso à internet, fazer stream de vídeo a diferentes velocidades. Clientes com conexões 3G low-speed podem receber uma versão com baixo bit-rate e baixa qualidade e os clientes com conexões com fibra podem receber uma versão de alta qualidade. O DASH também permite que um cliente se adapte à largura de banda disponível se a largura de banda ponto-a-ponto disponível mudar durante a sessão. Esta característica é particularmente importante para utilizadores mobile, que tipicamente veem a disponibilidade de banda flutuar conforme se movem.

Com o DASH, cada versão do vídeo é guardada no servidor HTTP com um URL diferente. O servidor HTTP também tem um ficheiro *manifest*, que fornece um link URL para cada versão e a sua taxa de bits. O cliente pede primeiro o ficheiro *manifest* descobrindo as várias versões. O cliente seleciona um chunk de cada vez ao especificar um URL e um intervalo de bytes numa mensagem HTTP GET request para cada chunk. Enquanto é feito o download dos chunks, o cliente também mede a largura de banda recebida e corre um algoritmo que determine a taxa para selecionar o próximo chunk a pedir. Naturalmente, se o cliente tem muito vídeo buffered e se a medida de banda recebida é alta ele vai escolher um chunk da versão high-bitrate. E naturalmente se o cliente tem

pouco video buffered e a medida de banda recebida é baixa, ele vai escolher o chunk da versão low-bitrate. Desta forma, o DASH permite ao cliente ter a liberdade de mudar entre diferentes níveis de qualidade.”.

Para este caso concreto, o DASH verificou a largura de banda através do ficheiro *"video-manifest.mpd"*, descobrindo qual o vídeo de resolução mais adequada. Assim, o DASH foi adaptando a resolução do video, tendo sempre em conta a qualidade de transmissão e de conexão.

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

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.

Nesta etapa, foram avaliados diferentes cenários de rede, considerando a transmissão de dados por unicast e multicast. Unicast é um método de endereçamento de pacotes que se destina a um único destinatário, sendo que a entrega é direcionada de forma simples, ou seja, ponto-a-ponto. Já o multicast envia dados para um grupo de destinatários, reduzindo assim a sobrecarga de tráfego na rede.

Para comparar ambos os cenários usamos o protocol hierarchy nas transmissões efetuadas. Com os dados obtidos, observamos que os dados enviados em unicast foram 185k e em multicast foram 170k. No entanto, os 185k dados enviados foram para apenas um cliente, enquanto que os 170k dados foram enviados para quatro clientes, o que realmente demonstra que o serviço multicast é mais eficiente.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	14486	100.0	12168238	195 k	0	0	0
Ethernet	100.0	14486	1.7	202804	3264	0	0	0
Internet Protocol Version 6	0.4	56	0.0	2240	36	0	0	0
User Datagram Protocol	0.0	2	0.0	16	0	0	0	0
Multicast Domain Name System	0.0	2	0.0	90	1	2	90	1
Open Shortest Path First	0.3	50	0.0	1800	28	50	1800	28
Internet Control Message Protocol v6	0.0	4	0.0	64	1	4	64	1
Internet Protocol Version 4	99.4	14404	2.4	288080	4636	0	0	0
User Datagram Protocol	97.7	14155	0.9	113240	1822	0	0	0
Data	97.5	14130	94.9	11545820	185 k	14130	11545820	185 k
ADwin configuration protocol	0.2	25	0.0	2400	38	25	2400	38
Open Shortest Path First	1.7	249	0.1	10956	176	249	10956	176
Address Resolution Protocol	0.2	26	0.0	728	11	26	728	11

Figure 26: Protocolo hierarchy na transmissão unicast

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	1064	100.0	888918	195 k	0	0	0
▼ Ethernet	100.0	1064	1.7	14896	3278	0	0	0
▼ Internet Protocol Version 6	0.1	1	0.0	40	8	0	0	0
Internet Control Message Protocol v6	0.1	1	0.0	16	3	1	16	3
▼ Internet Protocol Version 4	99.9	1063	2.4	21260	4679	0	0	0
▼ User Datagram Protocol	99.9	1063	1.0	8504	1871	0	0	0
▼ Session Announcement Protocol	0.7	7	0.3	2268	499	0	0	0
Session Description Protocol	0.7	7	0.2	2100	462	7	2100	462
▼ Real-time Transport Protocol	89.5	952	87.4	776673	170 k	0	0	0
MPV-ES	89.5	952	86.1	765249	168 k	952	765249	168 k
Real-time Transport Control Protocol	0.7	7	0.0	196	43	7	196	43
Data	9.1	97	7.3	65065	14 k	97	65065	14 k

Figure 27: Protocolo hierarchy na transmissão multicast

Uma grande vantagem do multicast é a sua escalabilidade, uma vez que é bem mais eficiente quando existem vários clientes na rede e à medida que o número destes aumenta a sobrecarga na rede permanece relativamente constante. Esta vantagem está diretamente relacionada com um melhor tráfego na rede, economizando banda larga.

Embora existam grandes vantagens no serviço multicast, há também algumas desvantagens. Uma delas é a complexidade de roteamento que pode ser mais difícil de gerir e manter. Há também dispositivos de rede que não possuem suporte completo para multicast, o que pode limitar a implementação em certos cenários. A eficiência do multicast é alcançada através da redução da flexibilidade, sendo que a capacidade de negociar parâmetros de transmissão e utilização com o emissor é limitada ou ausente.

3 Comentários Finais

Em modo de conclusão, neste trabalho prático aprofundamos o nosso conhecimento relativamente aos vários protocolos de *streaming* de dados. Deste modo, foi possível analisar o HTTP estático e o funcionamento do DASH, ambos sobre protocolo TCP. Também foi possível comparar os cenários de unicast e multicast e concluir que no geral o mais benéfico será o multicast.

4 Anexos

4.1 Eatapa 1 - Resultados dos streams nos hosts Jasmine, Bela e Monstro

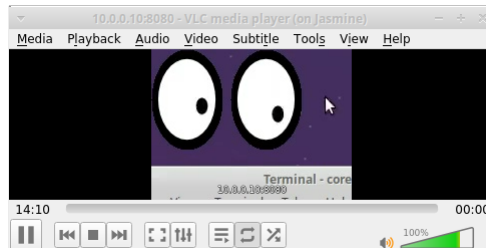


Figure 28: Resultado no host "Jasmine"

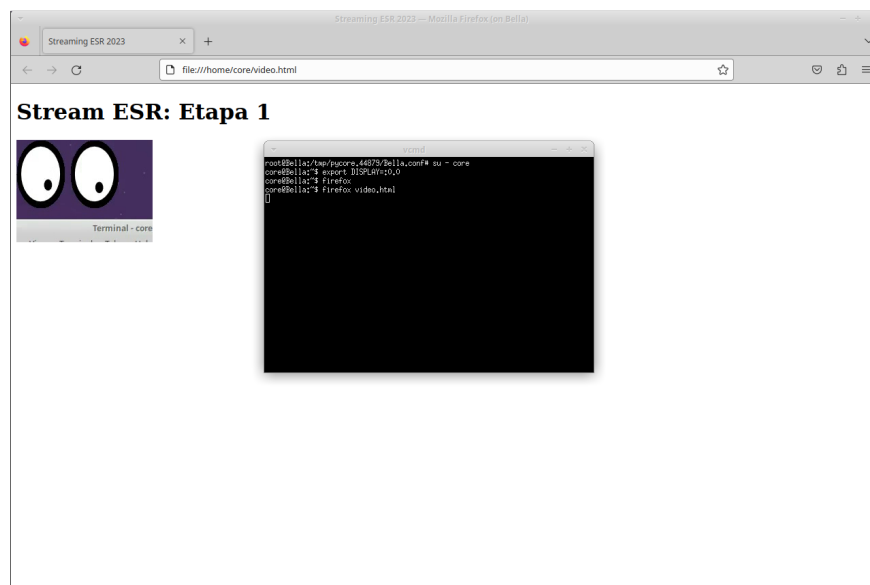


Figure 29: Resultado no host "Bela"

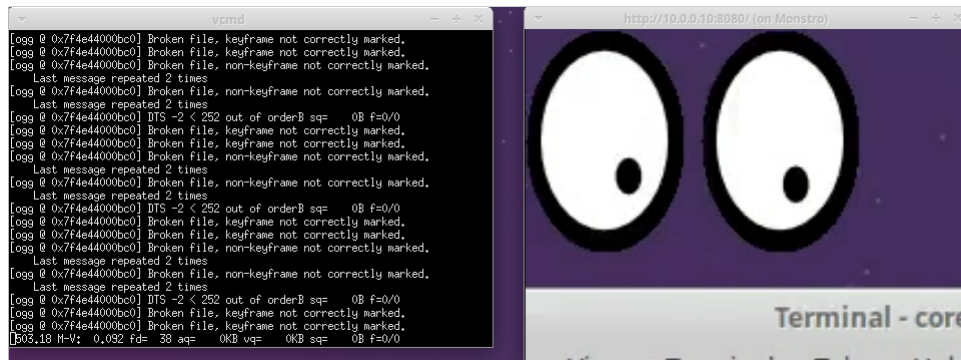


Figure 30: Resultado no host "Monstro"

4.2 Eatapa 2 - Resultados dos streams nos hosts Bela e Alladin

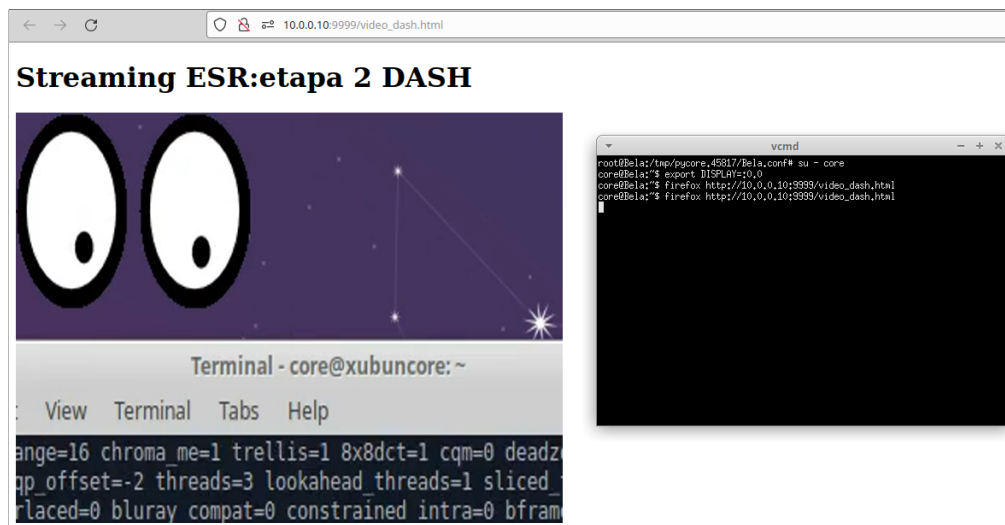


Figure 31: Resultado no host "Bela"

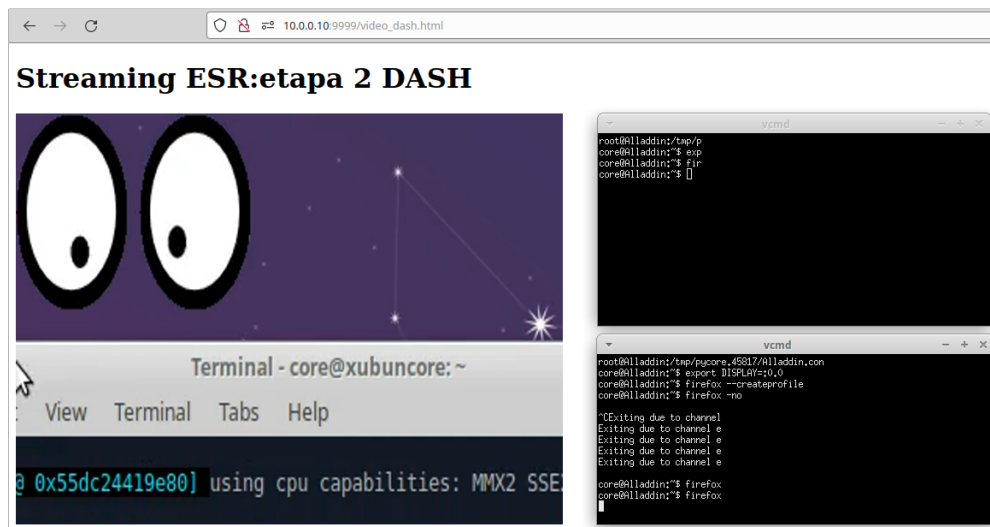


Figure 32: Resultado no host "Alladin"