

Engenharia de Serviços em Rede - TP1 - PL43

Afonso Xavier Cardoso Marques :: PG53601
Pedro Alexandre da Silva Oliveira :: PG55093
Vasco Rafael Barroso Gonçalves Rito :: PG55097

Outubro 2024



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) . .	10
2.3	Etapa 3 - Streaming RTP/RTCP unicast sobre UDP e multicast com anúncios SAP	15
3	Comentários Finais	19
4	Anexos	20
4.1	Eatapa 1 - Resultados dos streams nos hosts Jasmine, Bela e Monstro	20
4.2	Eatapa 2 - Resultados dos streams nos hosts Bela e Alladin . . .	20

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.

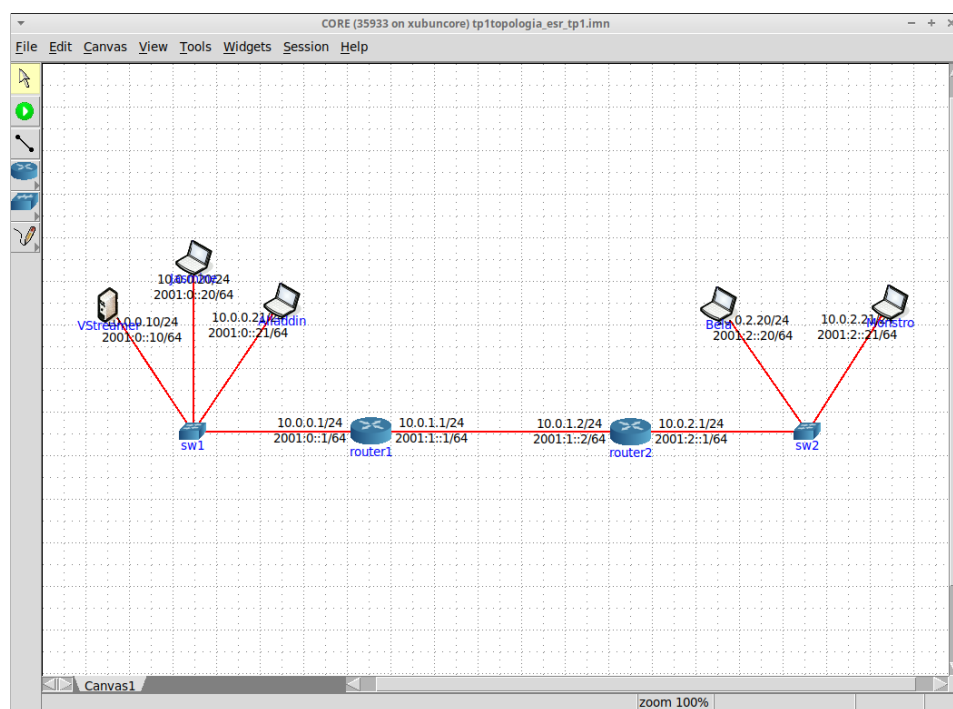


Figure 1: Topologia e os seus destaques

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 (usando o ffmpeg -i videoA.mp4 e/ou o próprio wireshark).

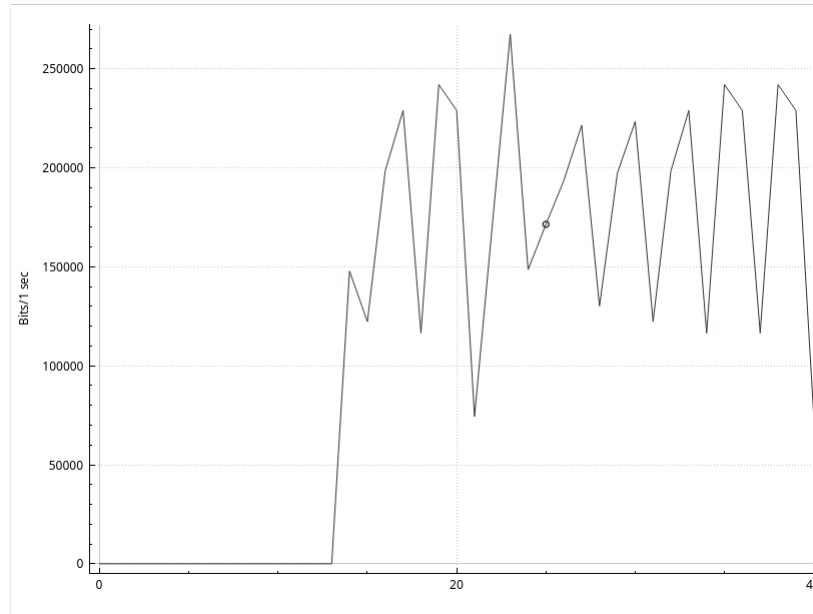


Figure 3: Variação de bps para o cliente Jasmine

Protocol	1 cliente	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame		100.0	2849	100.0	2005731	157 k	0	0	0
Ethernet		100.0	2849	2.0	39886	3124	0	0	0
Internet Protocol Version 6		0.4	10	0.0	400	31	0	0	0
Open Shortest Path First		0.4	10	0.0	360	28	10	360	28
Internet Protocol Version 4		99.4	2833	2.8	56660	4430	0	0	0
Transmission Control Protocol		97.6	2781	95.0	1905969	149 k	2780	1905803	149 k
Hypertext Transfer Protocol		0.0	1	0.0	134	10	1	134	10
Open Shortest Path First		1.8	52	0.1	2288	179	52	2288	179
Address Resolution Protocol		0.2	6	0.0	168	13	6	168	13

Figure 4: Protocol Hierarchy do tráfego com VStreamer para um cliente - Jasmine

Protocol	2 clientes	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame		100.0	1366	100.0	953147	196 k	0	0	0
Ethernet		100.0	1366	2.0	19124	3939	0	0	0
Internet Protocol Version 6		0.3	4	0.0	160	32	0	0	0
Open Shortest Path First		0.3	4	0.0	144	29	4	144	29
Internet Protocol Version 4		99.3	1356	2.8	27120	5586	0	0	0
Transmission Control Protocol		97.8	1336	95.0	905551	186 k	1334	905030	186 k
Hypertext Transfer Protocol		0.1	2	0.0	457	94	2	457	94
Open Shortest Path First		1.5	20	0.1	880	181	20	880	181
Address Resolution Protocol		0.4	6	0.0	168	34	6	168	34

Figure 5: Protocol Hierarchy do tráfego com VStreamer para dois clientes - Jasmine e Bela

Protocol	3 clientes	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame		100.0	7622	100.0	5443138	375 k	0	0	0
Ethernet		100.0	7622	2.0	106708	7355	0	0	0
Internet Protocol Version 6		0.1	11	0.0	440	30	0	0	0
Open Shortest Path First		0.1	11	0.0	396	27	11	396	27
Internet Protocol Version 4		99.7	7597	2.8	151940	10 k	0	0	0
Transmission Control Protocol		98.9	7538	95.2	5180666	357 k	7535	5179979	357 k
Hypertext Transfer Protocol		0.0	3	0.0	591	40	3	591	40
Open Shortest Path First		0.8	59	0.0	2596	178	59	2596	178
Address Resolution Protocol		0.2	14	0.0	392	27	14	392	27

Figure 6: Protocol Hierarchy do tráfego Com VStreamer para três clientes - Jasmine, Bela e Monstro

O aumento na taxa de bits por segundo entre o valor esperado (28 kbps) e o obtido nas três capturas poderá estar relacionado com o facto da transmissão ser feita em TCP, que acaba por adicionar muito overhead a cada pacote transmitido.

Na figura 7 conseguimos ver que a comunicação entre o servidor VStreamer e os três clientes, Jasmine, Bela e Monstro ocorreu maioritariamente em TCP. No que diz respeito ao uso do TCP para streaming de vídeo, este pode ser escolhido pois permite fazer controlo de congestão, através do ajuste da taxa de transmissão de acordo com as condições da rede; faz correção de erros através do envio de ACKs e da retransmissão de pacotes perdidos e garante uma entrega ordenada, evitando dados fora de sequência.

No.	Time	Source	Destination	Protocol	Length	Info
2626	55.217261172	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=546896 Win=68864 Len=0 TSval=1...
2627	55.217261786	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=548344 Win=68864 Len=0 TSval=1...
2628	55.217262254	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=549771 Win=67328 Len=0 TSval=1...
2629	55.217269800	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=261719 Ack=135 Win=65152 Len=1448 TSva...
2630	55.217269888	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=263167 Ack=135 Win=65152 Len=1448 TSva...
2631	55.217269966	10.0.0.10	10.0.2.21	TCP	1493	9090 → 47392 [PSH, ACK] Seq=264615 Ack=135 Win=65152 Len=1427...
2632	55.217288126	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=263167 Win=68864 Len=0 TSval=1...
2633	55.217288685	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=264615 Win=68864 Len=0 TSval=1...
2634	55.217289158	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=266042 Win=67328 Len=0 TSval=1...
2635	55.460896787	10.0.0.10	10.0.0.10	TCP	1514	9090 → 45472 [ACK] Seq=860225 Ack=135 Win=65152 Len=1448 TSva...
2636	55.460896219	10.0.0.10	10.0.0.10	TCP	1514	9090 → 45472 [ACK] Seq=867673 Ack=135 Win=65152 Len=1448 TSva...
2637	55.460896317	10.0.0.10	10.0.0.10	TCP	1353	9090 → 45472 [PSH, ACK] Seq=869121 Ack=135 Win=65152 Len=1287...
2638	55.461087141	10.0.0.10	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=867673 Win=74752 Len=0 TSval=2...
2639	55.461088501	10.0.0.10	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=869121 Win=74112 Len=0 TSval=2...
2640	55.461089168	10.0.0.10	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=870408 Win=73472 Len=0 TSval=2...
2641	55.461153025	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=549771 Ack=324 Win=64896 Len=1448 TSva...
2642	55.461153298	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=551219 Ack=324 Win=64896 Len=1448 TSva...
2643	55.461153394	10.0.0.10	10.0.2.20	TCP	1353	9090 → 60550 [PSH, ACK] Seq=552667 Ack=324 Win=64896 Len=1287...
2644	55.461210194	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=551219 Win=68864 Len=0 TSval=1...
2645	55.461211433	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=552667 Win=68864 Len=0 TSval=1...
2646	55.461212267	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=553954 Win=67456 Len=0 TSval=1...
2647	55.461226376	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=266042 Ack=135 Win=65152 Len=1448 TSva...
2648	55.461226574	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=267490 Ack=135 Win=65152 Len=1448 TSva...
2649	55.461226696	10.0.0.10	10.0.2.21	TCP	1353	9090 → 47392 [PSH, ACK] Seq=268938 Ack=135 Win=65152 Len=1287...
2650	55.461267013	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=267490 Win=68864 Len=0 TSval=1...
2651	55.461268023	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=268938 Win=68864 Len=0 TSval=1...
2652	55.461268856	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=270225 Win=67456 Len=0 TSval=1...
2653	55.953418211	10.0.0.10	10.0.0.10	TCP	1514	9090 → 45472 [ACK] Seq=870408 Ack=135 Win=65152 Len=1448 TSva...
2654	55.953418593	10.0.0.10	10.0.0.10	TCP	1514	9090 → 45472 [ACK] Seq=871856 Ack=135 Win=65152 Len=1448 TSva...
2655	55.953418663	10.0.0.10	10.0.0.10	TCP	1514	9090 → 45472 [ACK] Seq=873394 Ack=135 Win=65152 Len=1448 TSva...
2656	55.953418725	10.0.0.10	10.0.0.10	TCP	458	9090 → 45472 [PSH, ACK] Seq=874752 Ack=135 Win=65152 Len=392...

Figure 7: Captura dos pacotes - etapa 1

No entanto, num cenário onde a rede seja fraca e propicia a falhas, a retransmissão de pacotes perdidos pode afetar o desempenho de várias maneiras, tais como:

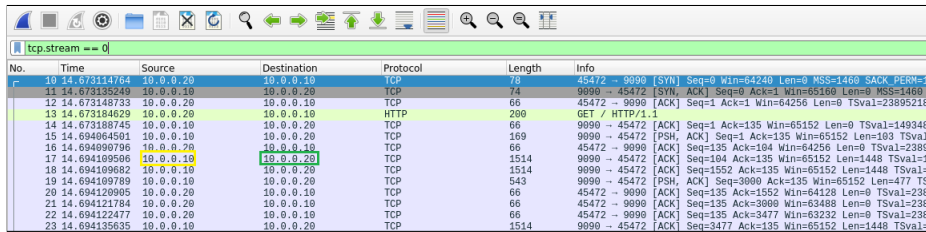
- Enquanto ocorre o reenvio do pacote perdido, o vídeo pode ficar parado, aguardando a chegada do pacote que falta. Isso introduz atrasos que

podem ser perceptíveis como pausas ou buffering no vídeo.

- O controlo de congestionamento pode levar à diminuição da largura de banda disponível para o stream. Como consequência, a qualidade do vídeo pode ser reduzida automaticamente (em streams adaptativos como o Youtube ou Netflix) ou ocorrer uma pausa para permitir o carregamento dos dados, causando uma experiência instável para o utilizador.

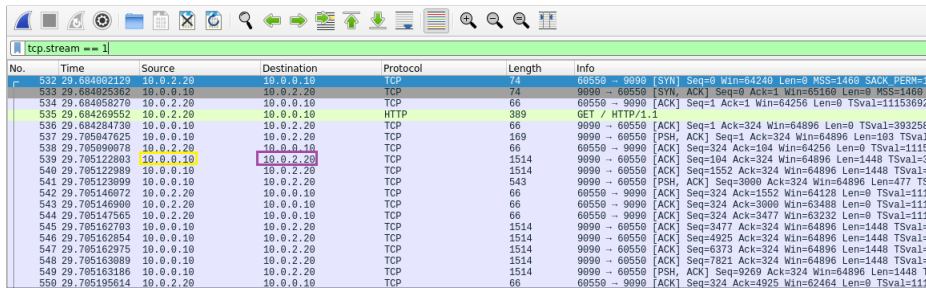
Relativamente ao número total de fluxos gerados recorremos ao filtro tcp.stream, que permite identificar as diferentes sessões TCP que ocorrem entre o servidor e os clientes numa rede. Como se trata de um cenário onde temos três clientes a receber um stream de um servidor, a expectativa é que o número de fluxos TCP a decorrer seja também de três, onde para o primeiro cliente, Jasmine, o fluxo seria o 0, para a Bela o 1 e para o Monstro o 2.

As seguintes capturas demonstram um segmento de cada um dos fluxos individuais.



No.	Time	Source	Destination	Protocol	Length	Info
10	14.673114764	10.0.0.10	10.0.0.10	TCP	74	45472 → 9090 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
11	14.673135249	10.0.0.10	10.0.0.10	TCP	74	9090 → 45472 [SYN, ACK] Seq=0 Ack=1 Win=65152 Len=0 MSS=1460
12	14.673148733	10.0.0.20	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=23895218
13	14.673184629	10.0.0.10	10.0.0.10	HTTP	200	GET / HTTP/1.1
14	14.673188745	10.0.0.10	10.0.0.20	TCP	66	9090 → 45472 [ACK] Seq=1 Ack=135 Win=65152 Len=0 TSval=149348
15	14.694064501	10.0.0.10	10.0.0.20	TCP	169	9090 → 45472 [PSH, ACK] Seq=1 Ack=135 Win=65152 Len=183 TSval=...
16	14.694090796	10.0.0.20	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=104 Win=64256 Len=0 TSval=2389...
17	14.694109506	10.0.0.10	10.0.0.20	TCP	1514	9090 → 45472 [ACK] Seq=104 Ack=135 Win=65152 Len=1448 TSval=1...
18	14.694109682	10.0.0.10	10.0.0.20	TCP	1514	9090 → 45472 [ACK] Seq=1552 Ack=135 Win=65152 Len=1448 TSval=...
19	14.694109789	10.0.0.10	10.0.0.20	TCP	543	9090 → 45472 [PSH, ACK] Seq=3000 Ack=135 Win=65152 Len=477 TS...
20	14.694120905	10.0.0.20	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=1552 Win=64128 Len=0 TSval=238...
21	14.694121784	10.0.0.20	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=3000 Win=63488 Len=0 TSval=238...
22	14.694122477	10.0.0.20	10.0.0.10	TCP	66	45472 → 9090 [ACK] Seq=135 Ack=3477 Win=63232 Len=0 TSval=238...
23	14.694135635	10.0.0.10	10.0.0.20	TCP	1514	9090 → 45472 [ACK] Seq=3477 Ack=135 Win=65152 Len=1448 TSval=...

Figure 8: Fluxo 0 - Jasmine



No.	Time	Source	Destination	Protocol	Length	Info
532	29.684902129	10.0.2.20	10.0.0.10	TCP	74	60550 → 9090 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
533	29.684925362	10.0.0.10	10.0.2.20	TCP	74	9090 → 60550 [SYN, ACK] Seq=0 Ack=1 Win=65152 Len=0 MSS=1460
534	29.684958276	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=11153692
535	29.684269552	10.0.2.20	10.0.0.10	HTTP	389	GET / HTTP/1.1
536	29.684284730	10.0.0.10	10.0.2.20	TCP	66	9090 → 60550 [ACK] Seq=1 Ack=324 Win=64896 Len=0 TSval=393258...
537	29.705047625	10.0.0.10	10.0.2.20	TCP	169	9090 → 60550 [PSH, ACK] Seq=1 Ack=324 Win=64896 Len=183 TSval=...
538	29.705090878	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=104 Win=64256 Len=0 TSval=1115...
539	29.705122803	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=104 Ack=324 Win=64896 Len=1448 TSval=3...
540	29.705122389	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=1552 Ack=324 Win=64896 Len=1448 TSval=...
541	29.705123099	10.0.0.10	10.0.2.20	TCP	543	9090 → 60550 [PSH, ACK] Seq=3000 Ack=324 Win=64896 Len=477 TS...
542	29.705146072	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=1552 Win=64128 Len=0 TSval=111...
543	29.705146900	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=3000 Win=63488 Len=0 TSval=111...
544	29.705147565	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=3477 Win=63232 Len=0 TSval=111...
545	29.705162703	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=3477 Ack=324 Win=64896 Len=1448 TSval=...
546	29.705162854	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=4925 Ack=324 Win=64896 Len=1448 TSval=...
547	29.705162975	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=6373 Ack=324 Win=64896 Len=1448 TSval=...
548	29.705163089	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [ACK] Seq=7821 Ack=324 Win=64896 Len=1448 TSval=...
549	29.705163186	10.0.0.10	10.0.2.20	TCP	1514	9090 → 60550 [PSH, ACK] Seq=9269 Ack=324 Win=64896 Len=1448 T...
550	29.705195614	10.0.2.20	10.0.0.10	TCP	66	60550 → 9090 [ACK] Seq=324 Ack=4925 Win=62464 Len=0 TSval=111...

Figure 9: Fluxo 1 - Bela

No.	Time	Source	Destination	Protocol	Length	Info
1423	43.391915075	10.0.2.21	10.0.0.10	TCP	74	47392 → 9090 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
1424	43.391944547	10.0.0.10	10.0.2.21	TCP	74	9090 → 47392 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
1425	43.391967971	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=15906113
1426	43.392066108	10.0.2.21	10.0.0.10	HTTP	200	GET / HTTP/1.1
1427	43.392099195	10.0.0.10	10.0.2.21	TCP	66	9090 → 47392 [ACK] Seq=1 Ack=135 Win=65152 Len=0 TSval=392141
1428	43.412296146	10.0.0.10	10.0.2.21	TCP	169	9090 → 47392 [PSH, ACK] Seq=1 Ack=135 Win=65152 Len=103 TSval=...
1429	43.412339676	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=104 Win=64256 Len=0 TSval=15906
1430	43.412354124	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=104 Ack=135 Win=65152 Len=1448 TSval=3...
1431	43.412354248	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=1552 Ack=135 Win=65152 Len=1448 TSval=...
1432	43.412354326	10.0.0.10	10.0.2.21	TCP	543	9090 → 47392 [PSH, ACK] Seq=3080 Ack=135 Win=65152 Len=477 TS...
1433	43.412372450	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=1552 Win=64128 Len=0 TSval=159...
1434	43.412373228	10.0.0.10	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=3080 Win=63488 Len=0 TSval=159...
1435	43.412373679	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=3477 Win=63232 Len=0 TSval=159...
1436	43.412394326	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=3477 Ack=135 Win=65152 Len=1448 TSval=...
1437	43.412394450	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=4925 Ack=135 Win=65152 Len=1448 TSval=...
1438	43.412394533	10.0.0.10	10.0.2.21	TCP	1514	9090 → 47392 [ACK] Seq=6373 Ack=135 Win=65152 Len=1448 TSval=...
1439	43.412394620	10.0.0.10	10.0.2.21	TCP	458	9090 → 47392 [PSH, ACK] Seq=7821 Ack=135 Win=65152 Len=392 TS...
1440	43.412418250	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=4925 Win=64128 Len=0 TSval=159...
1441	43.412418906	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=6373 Win=63488 Len=0 TSval=159...
1442	43.412419377	10.0.2.21	10.0.0.10	TCP	66	47392 → 9090 [ACK] Seq=135 Ack=7821 Win=62848 Len=0 TSval=159...

Figure 10: Fluxo 2 - Monstro

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

Figure 11: Fluxo 3 - Não existe fluxo

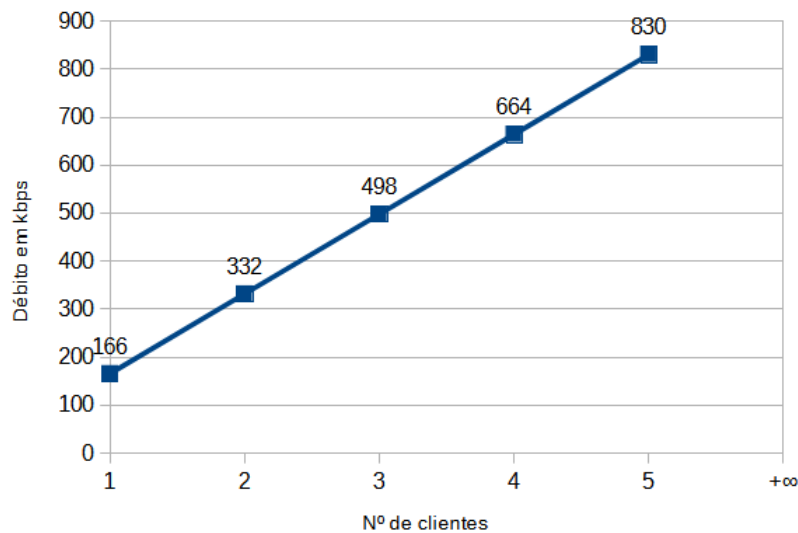
Para calcular o débito (D), podemos multiplicar a bitrate (b) do vídeo pelo número de clientes que acedem ao stream (N). A bitrate do vídeo foi obtida consultando o fluxo de stream de cada um dos clientes e calculando a média.

bps_jasmine = 177 kbps
 bps_bela = 143 kbps
 bps_monstro = 178 kbps

$$b = \frac{bps_jasmine + bps_bela + bps_monstro}{3} = 166 \text{ kbps}$$

$$D = b \times N = 166 \text{ kbps} \times N$$

O que permite construir o seguinte gráfico, onde fica bastante claro que a largura de banda aumenta linearmente conforme o número de clientes a assistir ao stream.



Por fim, a escalabilidade para 1000 ou 10000 utilizadores, da maneira como está implementado, obrigaria o servidor a enviar uma cópia do vídeo para cada um desses utilizadores, o que iria aumentar linearmente o débito da saída do servidor conforme o número de utilizadores aumentava. Com um número elevado de utilizadores e as ligações serem feitas utilizando o protocolo TCP, o overhead associado ao controlo de congestionamento e correção de erros poderia gerar atrasos significativos na transmissão.

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

Questão 2: Utilize o wireshark para determinar 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. Explique como obteve esta informação.

Nesta segunda questão, foi pedido para determinar a largura de banda necessária para que o cliente streaming consiga receber o vídeo no firefox.

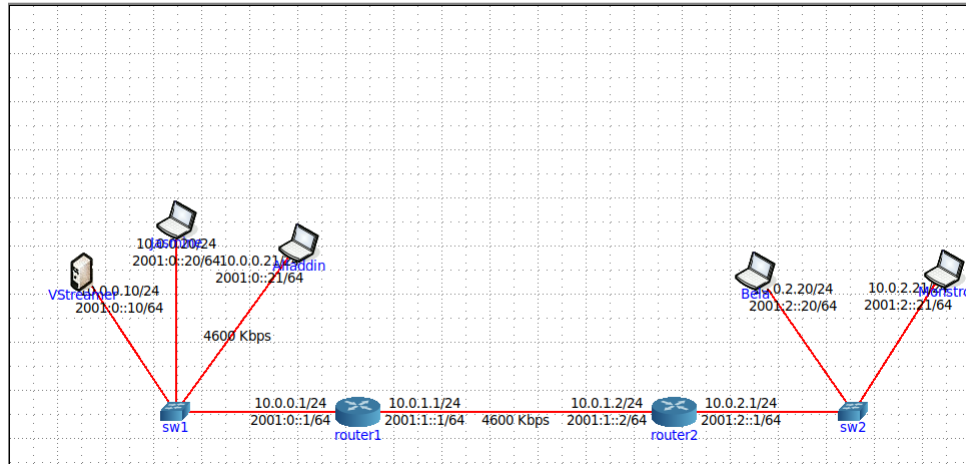


Figure 12: Topologia com links limitados a 4600kbps

Dessa forma, e após limitar os nós de ligação pretendidos, chegou-se à conclusão de que apenas uma largura de banda de 4600 kbps conseguia transmitir a resolução de 960x720 consistentemente, enquanto o manifest especifica que essa resolução só requer 509.1 kbps.

```
<Representation id="3" mimeType="video/mp4" codecs="avc3.64001f" width="960" height="720" frameRate="11456/384" sar="1:1" startWithSAP="0" bandwidth="509086">
  <BaseURL>videoB_960_720_1000k_dash.mp4</BaseURL>
  <SegmentList timescale="11456" duration="5750">
    <SegmentURL mediaRange="927-24931" indexRange="927-970"/>
    <SegmentURL mediaRange="24932-48807" indexRange="24932-24975"/>
    <SegmentURL mediaRange="48808-73448" indexRange="48808-48851"/>
    <SegmentURL mediaRange="73449-132819" indexRange="73449-73492"/>
    <SegmentURL mediaRange="132820-153706" indexRange="132820-132863"/>
    <SegmentURL mediaRange="153707-183534" indexRange="153707-153750"/>
    <SegmentURL mediaRange="183535-255704" indexRange="183535-183578"/>
    <SegmentURL mediaRange="255705-279255" indexRange="255705-255748"/>
    <SegmentURL mediaRange="279256-301153" indexRange="279256-279299"/>
    <SegmentURL mediaRange="301154-302891" indexRange="301154-301197"/>
  </SegmentList>
```

Figure 13: Largura necessária descrita no video_manifest.mpd

Com isso, e comparando com o manifest, conseguimos notar que há algumas possíveis causas para o sucedido, tais como:

- Oscilações de rede, como o vídeo é muito pequeno, qualquer oscilação na largura de banda pode causar o reprodutor a optar por uma qualidade inferior, o que pode explicar por que apenas uma largura de banda tão alta como 4600 kbps consegue garantir a qualidade máxima de forma consistente.
- Buffering, o atributo minBufferTime="PT1.500S" sugere que o reprodutor precisa de acumular pelo menos 1.5 segundos de vídeo antes de começar a reproduzir. Se o sistema tiver dificuldades em manter esse buffer com menos largura de banda, ele pode estar a alternar para uma qualidade inferior para evitar pausas na reprodução.

```

-<MPD minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H0M4.760S"
-<ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
  <Title>video_manifest.mpd generated by GPAC</Title>

```

- Overhead protocolar, a largura de banda total usada durante a transmissão será maior devido ao overhead dos protocolos de transporte (como TCP ou HTTP).

Ao utilizarmos o Streaming adaptativo sobre HTTP (MPEG-DASH), teremos várias versões do vídeo com diferentes bitrates. Estes vídeos estão divididos em vários segmentos, o que pode ser visto no ficheiro de manifesto. Periodicamente o cliente vai verificando a largura de banda. através de pedidos HTTP ao servidor. A pilha protocolar corresponde à camada da aplicação por HTTP.

No.	Time	Source	Destination	Protocol	Length	Info
13	12.961703443	10.0.2.20	10.0.0.10	HTTP	425	GET /video_dash.html HTTP/1.1
15	12.961921299	10.0.0.10	10.0.2.20	HTTP	563	HTTP/1.1 200 OK (text/html)
23	16.189785555	10.0.2.20	10.0.0.10	HTTP	379	GET /favicon.ico HTTP/1.1
25	16.196146326	10.0.0.10	10.0.2.20	HTTP	741	HTTP/1.1 404 Not Found (text/html)
35	17.729422075	10.0.2.20	10.0.0.10	HTTP	369	GET /video_manifest_init.mpd HTTP/1.1
38	17.729564713	10.0.0.10	10.0.2.20	MP4	1060	
43	17.887431962	10.0.2.20	10.0.0.10	HTTP	399	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
351	18.337072796	10.0.0.10	10.0.2.20	MP4	531	
405	18.656521115	10.0.2.20	10.0.0.10	HTTP	401	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
730	19.099871385	10.0.0.10	10.0.2.20	MP4	531	
754	19.606584703	10.0.2.20	10.0.0.10	HTTP	401	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
1082	20.053045412	10.0.0.10	10.0.2.20	MP4	531	
1106	20.353162935	10.0.2.20	10.0.0.10	HTTP	402	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
1436	20.865036121	10.0.0.10	10.0.2.20	MP4	531	
1457	21.039155353	10.0.2.20	10.0.0.10	HTTP	403	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
1792	21.483756926	10.0.0.10	10.0.2.20	MP4	531	
1816	21.872213354	10.0.2.20	10.0.0.10	HTTP	403	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
2143	22.325694238	10.0.0.10	10.0.2.20	MP4	531	
2185	22.465801581	10.0.2.20	10.0.0.10	HTTP	403	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
2496	22.936868254	10.0.0.10	10.0.2.20	MP4	531	
2521	23.127585512	10.0.2.20	10.0.0.10	HTTP	403	GET /videoB_960_720_1000k_dash.mpd HTTP/1.1
2851	23.588445913	10.0.0.10	10.0.2.20	MP4	531	

Figure 14: Captura pacotes HTTP

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).

Questão 3: Compare a largura de banda medida na questão anterior com a que é disponibilizada pelo ffmpeg. Qual é a razão para a diferença entre as duas?

De forma a fazer a comparação, recorreremos à ferramenta ffmpeg no vídeo de mais alta resolução, videoB_960_720_1000k.mp4, onde obtivemos o valor de 502 kbps para a largura de banda.

Como pode ser visto na foto abaixo limitamos o link entre o Switch 1 e o router 1 a 157.72 kpbs.

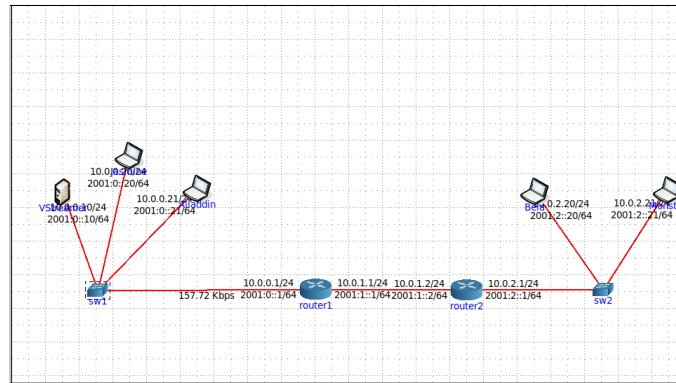


Figure 17: Topologia com limitação

No.	Time	Source	Destination	Protocol	Length	Info
158	16.6687836	10.0.0.10	10.0.2.20	HTTP	741	HTTP/1.1 404 Not Found (text/html)
166	26.3232058	10.0.2.20	10.0.0.10	HTTP	476	GET /video_manifest.mpd HTTP/1.1
168	26.32338978	10.0.0.10	10.0.2.20	HTTP	273	HTTP/1.1 304 Not Modified
293	38.853643343	10.0.2.20	10.0.0.10	HTTP	419	GET /video_manifest_init.mp4 HTTP/1.1
295	38.853941866	10.0.0.10	10.0.2.20	HTTP	257	HTTP/1.1 304 Not Modified
510	38.774871584	10.0.2.20	10.0.0.10	HTTP	399	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
1390	69.533259761	10.0.2.20	10.0.0.10	HTTP	399	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
1446	70.494757487	10.0.2.20	10.0.0.10	HTTP	398	GET /video0_720_540_1000k_dash.mpd HTTP/1.1
1593	89.975568619	10.0.2.20	10.0.0.10	HTTP	399	GET /video0_360_226_200k_dash.mpd HTTP/1.1
1757	82.21698488	10.0.0.10	10.0.2.20	MP4	646	
1791	84.441063453	10.0.2.20	10.0.0.10	HTTP	400	GET /video0_360_226_200k_dash.mpd HTTP/1.1
1855	85.750057502	10.0.0.10	10.0.2.20	MP4	646	
1890	87.863744672	10.0.2.20	10.0.0.10	HTTP	400	GET /video0_360_226_200k_dash.mpd HTTP/1.1
1954	89.213516287	10.0.0.10	10.0.2.20	MP4	646	
1995	91.280557500	10.0.2.20	10.0.0.10	HTTP	400	GET /video0_360_226_200k_dash.mpd HTTP/1.1
2062	92.657518395	10.0.0.10	10.0.2.20	MP4	646	

Figure 18: Captura de pacotes relativos ao cliente Bela

Na captura podemos ver o portátil da Bela a fazer dois pedidos GET do vídeo de maior resolução, as quais não recebeu nenhuma resposta com o chunk de vídeo, depois fez um GET do vídeo de resolução intermédia mas os restantes foram todos GET do vídeo de menor resolução pois era o único que a sua ligação permitia receber a transmissão haver interrupções.

No.	Time	Source	Destination	Protocol	Length	Info
2106	116.990542932	10.0.0.21	10.0.0.10	HTTP	475	GET /video_dash.html HTTP/1.1
2108	116.990682390	10.0.0.10	10.0.0.21	HTTP	272	HTTP/1.1 304 Not Modified
2115	118.085837792	10.0.0.21	10.0.0.10	HTTP	379	GET /favicon.ico HTTP/1.1
2117	118.085942686	10.0.0.10	10.0.0.21	HTTP	741	HTTP/1.1 404 Not Found (text/html)
2124	119.090428426	10.0.0.21	10.0.0.10	HTTP	476	GET /video_manifest.mpd HTTP/1.1
2126	119.090565348	10.0.0.10	10.0.0.21	HTTP	273	HTTP/1.1 304 Not Modified
2132	119.976154822	10.0.0.21	10.0.0.10	HTTP	419	GET /video_manifest_init.mp4 HTTP/1.1
2134	119.976235065	10.0.0.10	10.0.0.21	HTTP	257	HTTP/1.1 304 Not Modified
2148	120.003817037	10.0.0.21	10.0.0.10	HTTP	399	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
2478	120.024378396	10.0.0.10	10.0.0.21	MP4	531	
2485	120.055698569	10.0.0.21	10.0.0.10	HTTP	401	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
2739	120.657398115	10.0.0.10	10.0.0.21	MP4	531	
2748	120.060488063	10.0.0.21	10.0.0.10	HTTP	401	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
2984	120.072218366	10.0.0.10	10.0.0.21	MP4	531	
2991	120.088878909	10.0.0.21	10.0.0.10	HTTP	402	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
3245	120.091253666	10.0.0.10	10.0.0.21	MP4	531	
3253	120.104691107	10.0.0.21	10.0.0.10	HTTP	403	GET /video0_960_720_1000k_dash.mpd HTTP/1.1
3496	120.100240060	10.0.0.10	10.0.0.21	MP4	531	

Figure 19: Captura de pacotes relativos ao cliente Alladin

Na imagem acima podemos ver que o portátil do Alladin apresentou sempre o vídeo de maior resolução pois não tinha nenhuma limitação na ligação.

Questão 5: Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado e comparando o modelo de streaming com o que foi utilizado na Questão 1.

Como foi anteriormente referido o DASH (Dynamic Adaptive Streaming over HTTP) consiste na representação de diferentes versões de bitrate de um ficheiro de multimédia. Cada uma dessas representações está dividida em diferentes segmentos/chunks de vídeo. Toda essa informação encontra-se armazenada num ficheiro de manifesto que fornece os URL's para chunks diferentes. Assim, inicialmente, os clientes fazem o pedido ao servidor desse ficheiro de manifesto e, periodicamente, avaliam a largura de banda disponível. Deste modo, o cliente irá solicitando ao servidor o chunk que tenha um bitrate máximo para as condições de rede disponíveis do seu lado. Logo, a qualquer momento a representação do vídeo que o cliente recebe poderá mudar, pelo que a qualidade do vídeo irá se ajustar/adaptar em conformidade com a largura de banda no momento.

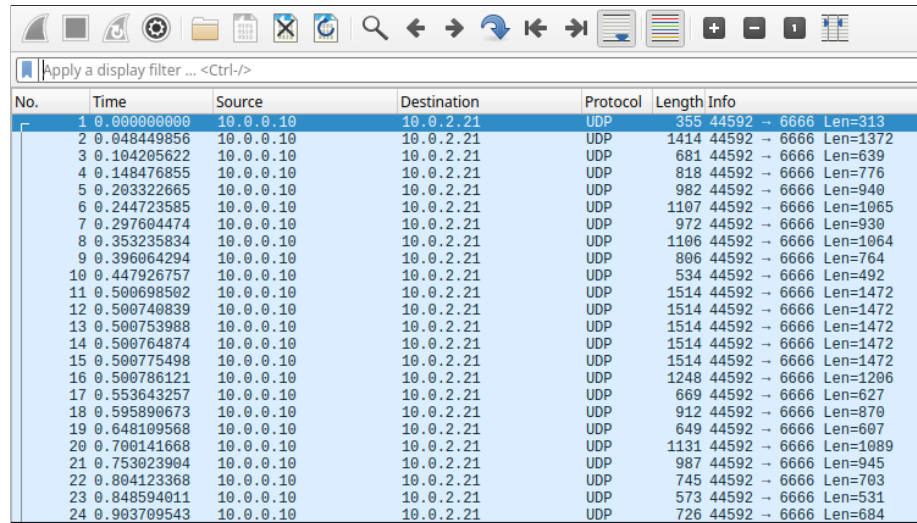
O modelo de streaming utilizado na questão 1 é o Streaming HTTP Simple (Sem Adaptação Dinâmica de Débito). Conforme foi dito acima, no DASH, o conteúdo é segmentado em pequenos pedaços e codificado em vários bitrates, sendo a qualidade do vídeo ajustada automaticamente conforme as condições de rede do cliente, enquanto no Streaming HTTP Simple o conteúdo é transmitido num único bitrate independentemente das condições da rede. O DASH oferece uma experiência ao cliente mais suave, minimizando o buffering em redes mais lentas, por outro lado o Streaming de HTTP Simple é mais simples de implementar.

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

Questão 6: 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 também para os cenários de 1000 e 10000 clientes.

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.

No nível de rede, o unicast acaba por ser um tipo de transmissão mais desvantajosa, uma vez que, caso seja necessário transmitir para mais do que um destino, haverá a necessidade de existirem N fluxos, uma vez que a ligação é de um para um, ao contrário do multicast que apenas envia uma vez os dados e, posteriormente, os switches e routers replicam para os vários destinatários. Podemos comprovar estes factos, com a observação das seguintes capturas do wireshark em cenários de unicast e multicast.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.2.21	UDP	355	44592 → 6666 Len=313
2	0.048449856	10.0.0.10	10.0.2.21	UDP	1414	44592 → 6666 Len=1372
3	0.104205622	10.0.0.10	10.0.2.21	UDP	681	44592 → 6666 Len=639
4	0.148476855	10.0.0.10	10.0.2.21	UDP	818	44592 → 6666 Len=776
5	0.203322665	10.0.0.10	10.0.2.21	UDP	982	44592 → 6666 Len=940
6	0.244723585	10.0.0.10	10.0.2.21	UDP	1107	44592 → 6666 Len=1065
7	0.297604474	10.0.0.10	10.0.2.21	UDP	972	44592 → 6666 Len=930
8	0.353235834	10.0.0.10	10.0.2.21	UDP	1106	44592 → 6666 Len=1064
9	0.396064294	10.0.0.10	10.0.2.21	UDP	806	44592 → 6666 Len=764
10	0.447926757	10.0.0.10	10.0.2.21	UDP	534	44592 → 6666 Len=492
11	0.500698502	10.0.0.10	10.0.2.21	UDP	1514	44592 → 6666 Len=1472
12	0.500740839	10.0.0.10	10.0.2.21	UDP	1514	44592 → 6666 Len=1472
13	0.500753988	10.0.0.10	10.0.2.21	UDP	1514	44592 → 6666 Len=1472
14	0.500764874	10.0.0.10	10.0.2.21	UDP	1514	44592 → 6666 Len=1472
15	0.500775498	10.0.0.10	10.0.2.21	UDP	1514	44592 → 6666 Len=1472
16	0.500786121	10.0.0.10	10.0.2.21	UDP	1248	44592 → 6666 Len=1206
17	0.553643257	10.0.0.10	10.0.2.21	UDP	669	44592 → 6666 Len=627
18	0.595890673	10.0.0.10	10.0.2.21	UDP	912	44592 → 6666 Len=870
19	0.648109568	10.0.0.10	10.0.2.21	UDP	649	44592 → 6666 Len=607
20	0.709141668	10.0.0.10	10.0.2.21	UDP	1131	44592 → 6666 Len=1089
21	0.753023904	10.0.0.10	10.0.2.21	UDP	987	44592 → 6666 Len=945
22	0.804123368	10.0.0.10	10.0.2.21	UDP	745	44592 → 6666 Len=703
23	0.848594011	10.0.0.10	10.0.2.21	UDP	573	44592 → 6666 Len=531
24	0.903709543	10.0.0.10	10.0.2.21	UDP	726	44592 → 6666 Len=684

Figure 20: Captura unicast

No cenário em unicast o vídeo é enviado diretamente do VStreamer para o Monstro, esta transmissão é feita sobre UDP (connectionless). Em ambos os cenários de transmissão é da responsabilidade da camada de aplicação fazer o controle de fluxos, de erros e de congestionamento

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	224.0.0.224	UDP	907	52259 → 6666 Len=865
2	0.056078017	10.0.0.10	224.0.0.224	UDP	914	52259 → 6666 Len=872
3	0.107947973	10.0.0.10	224.0.0.224	UDP	992	52259 → 6666 Len=950
4	0.157772504	10.0.0.10	224.0.0.224	UDP	733	52259 → 6666 Len=691
5	0.202418295	10.0.0.10	224.0.0.224	UDP	1397	52259 → 6666 Len=1355
6	0.254670489	10.0.0.10	224.0.0.224	UDP	677	52259 → 6666 Len=635
7	0.299974800	10.0.0.10	224.0.0.224	UDP	817	52259 → 6666 Len=775
8	0.353320446	10.0.0.10	224.0.0.224	UDP	1514	52259 → 6666 Len=1472
9	0.354268144	10.0.0.10	224.0.0.224	UDP	1514	52259 → 6666 Len=1472
10	0.354295618	10.0.0.10	224.0.0.224	UDP	1514	52259 → 6666 Len=1472
11	0.354309458	10.0.0.10	224.0.0.224	UDP	1514	52259 → 6666 Len=1472
12	0.354322039	10.0.0.10	224.0.0.224	UDP	1147	52259 → 6666 Len=1105
13	0.398315680	10.0.0.10	224.0.0.224	UDP	1113	52259 → 6666 Len=1071
14	0.452197656	10.0.0.10	224.0.0.224	UDP	976	52259 → 6666 Len=934
15	0.507743113	10.0.0.10	224.0.0.224	UDP	1071	52259 → 6666 Len=1029
16	0.554143178	10.0.0.10	224.0.0.224	UDP	811	52259 → 6666 Len=769
17	0.607206949	10.0.0.10	224.0.0.224	UDP	519	52259 → 6666 Len=477
18	0.657274755	10.0.0.10	224.0.0.224	UDP	918	52259 → 6666 Len=876
19	0.714018033	10.0.0.10	224.0.0.224	UDP	678	52259 → 6666 Len=636
20	0.752726953	10.0.0.10	224.0.0.224	UDP	932	52259 → 6666 Len=890
21	0.798052312	10.0.0.10	224.0.0.224	UDP	652	52259 → 6666 Len=610
22	0.856988438	10.0.0.10	224.0.0.224	UDP	1386	52259 → 6666 Len=1344
23	0.906574146	10.0.0.10	224.0.0.224	UDP	984	52259 → 6666 Len=942
24	0.960552679	10.0.0.10	224.0.0.224	UDP	1514	52259 → 6666 Len=1472

Figure 21: Captura multicast

Por outro lado, no cenário com multicast, este apenas envia para um grupo multicast (neste caso o 224.0.0.224 porta 6666) em UDP, onde qualquer host consegue aceder desde que se encontre à escuta para este grupo.

Para comparar ambos os cenários no que diz respeito aos débitos de transmissões efetuadas, observamos que os dados enviados em unicast foram chegando a uma taxa de 211 kbps enquanto que em multicast o valor foi de 210 kbps

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	44592	10.0.2.21	6666	1,738	1641 k	1,738	1641 k	0	0	0.000000	61.9998	211 k	0
10.0.0.10	44593	10.0.2.21	6667	12	840	12	840	0	0	2.952776	55.2500	121	0

Figure 22: Taxa de transmissão em unicast

Ethernet - 4	IPv4 - 2	IPv6 - 2	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	
10.0.0.10	52259	224.0.0.224	6666	1,058	997 k	1,058	997 k	0	0	0.000000	37.8565	210 k	0	
10.0.0.10	49259	224.2.127.254	9875	8	2928	8	2928	0	0	1.507663	35.2423	664	0	
10.0.0.10	52260	224.0.0.224	6667	8	560	8	560	0	0	1.648751	35.1581	127	0	

Figure 23: Taxa de transmissão em multicast

Relativamente ao unicast, este é mais fácil de implementar, no entanto existe um desperdício de largura de banda.

No que diz respeito ao multicast, uma grande vantagem 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.

Em jeito de conclusão, a escolha de um ou outro, acaba por depender do número de clientes e das necessidades do sistema. Por exemplo, num sistema com 1000 ou 10000 clientes, a escolha mais acertada e menos taxadora seria o multicast.

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 24: Resultado nos três hosts

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

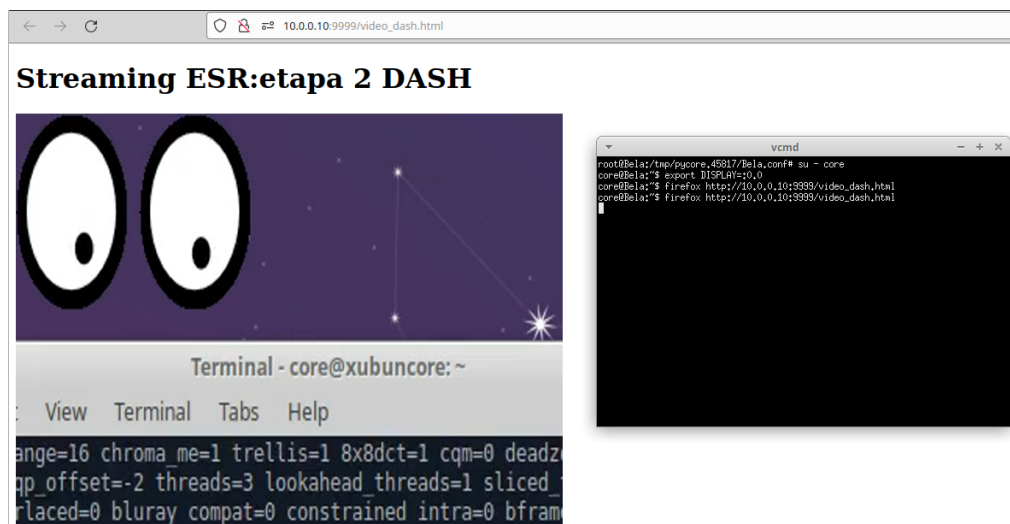


Figure 25: Resultado no host "Bela"

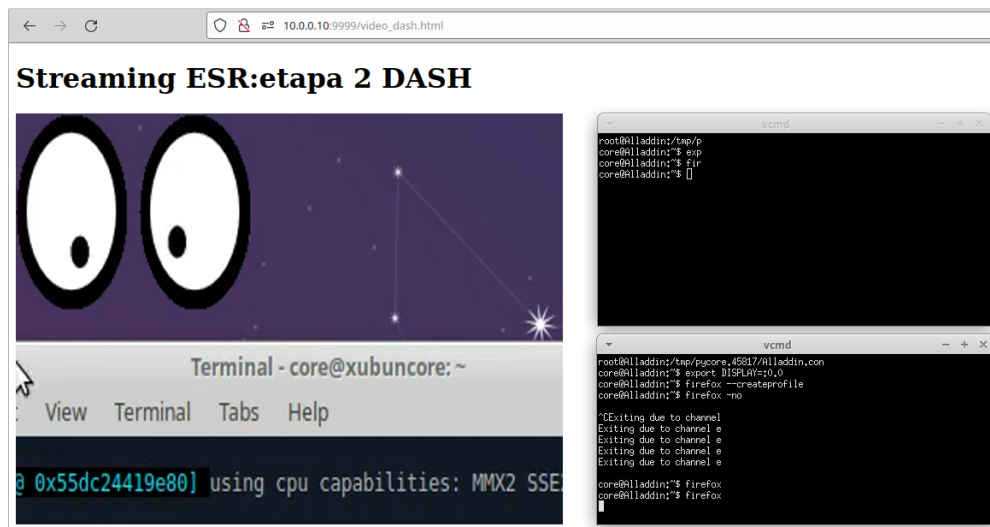


Figure 26: Resultado no host "Alladin"