

TP1: – Streaming de áudio e vídeo a pedido e em tempo real

Cláudia Silva^[a93177], David Duarte^[pg50315], and Laura Rodrigues^[pg50542]

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
{a93177,pg50315,pg50542}@alunos.uminho.pt

1 Questões e Respostas

1.1 Streaming HTTP simples sem adaptação dinâmica de débito

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

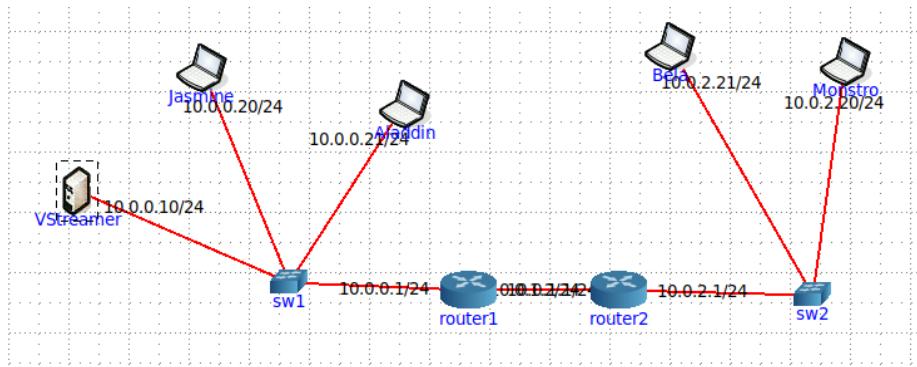


Fig. 1. Topologia criada

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).

A taxa em bps esperada pode ser consultada através do comando `ffmpeg -i videoA.mp4`, como podemos ver pela figura 2 abaixo.

```
ffmpeg version 4.2.7-0ubuntu0.1 Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-ubuntu1-20.04.1)
  configuration: --prefix=/usr --extra-version=0ubuntu1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl --disable-stripping --enable-avresample --disable-filter=resample --enable-avisynth --enable-e-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libopus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared

libavutil      56. 31.100 / 56. 31.100
libavcodec     58. 54.100 / 58. 54.100
libavformat    58. 29.100 / 58. 29.100
libavdevice    58.  8.100 / 58.  8.100
libavfilter     7. 57.100 / 7. 57.100
libavresample   4.  0. 0 / 4.  0. 0
libswscale      5.  5.100 / 5.  5.100
libswresample   3.  5.100 / 3.  5.100
libpostproc    55.  5.100 / 55.  5.100

Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoA.mp4':
  Metadata:
    major_brand : isom
    minor_version : 512
    compatible_brands: isomiso2avc1mp41
Duration: 00:00:12.10, start: 0.000000, bitrate: 13 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 160x100, 10 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
  Metadata:
    handler_name : VideoHandler
```

Fig. 2. Taxa em bps esperada

Sabemos assim que a taxa apresentada é de 10 kbps, no entanto, ao capturarmos o tráfego no link de saída do servidor com 1 cliente (VLC) - figura 3 - podemos observar que a taxa realmente necessária para transmitir o vídeo foi 21kbps. A diferença de valores ocorre devido à ocorrência de perdas de transmissão, valor visível na figura 3 em Malformed Packet.

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 ffplay) é 54kbps e 106kbps respectivamente.

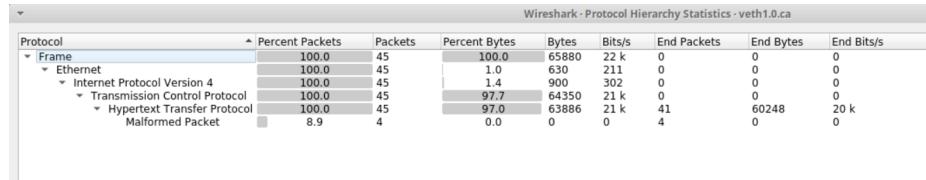


Fig. 3. Protocol Hierarchy do tráfego com servidor e Jasmin

Wireshark - Protocol Hierarchy Statistics - vnet1.v1									
Protocol		Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame		100.0	424	100.0	279330	176 k	0	0	0
Ethernet		100.0	424	2.1	5936	3741	0	0	0
Internet Protocol Version 6		0.5	2	0.0	80	50	0	0	0
Open Shortest Path First		0.2	1	0.0	36	22	1	36	22
Internet Control Message Protocol v6		0.2	1	0.0	16	10	1	16	10
Internet Protocol Version 4		99.5	422	3.0	8440	5320	0	0	0
Transmission Control Protocol		98.1	416	94.7	264558	166 k	356	177976	112 k
HyperText Transfer Protocol		14.2	60	31.0	86614	54 k	54	80080	50 k
Malformed Packet		1.4	6	0.0	0	0	6	0	0
Open Shortest Path First		1.4	6	0.1	264	166	6	264	166

Fig. 4. Protocol Hierarchy do tráfego com servidor e Jasmin + Bela

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	158	100.0	101412	354 k	0	0	0
Ethernet	100.0	158	2.2	2212	7728	0	0	0
Internet Protocol Version 4	100.0	158	3.1	3160	11 k	0	0	0
Transmission Control Protocol	98.7	156	94.6	95952	335 k	135	64872	226 k
Hypertext Transfer Protocol	13.3	21	30.0	30408	106 k	21	30408	106 k
Open Shortest Path First	1.3	2	0.1	88	307	2	88	307

Fig. 5. Protocol Hierarchy do tráfego Com servidor e Jasmin + 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).

Fig. 6. Tráfego com servidor e Jasmin (1 cliente)

Fig. 7. Tráfego com servidor e Jasmin + Bela (2 clientes)

No.	Time	Source	Destination	Protocol	Length: Info
1187	25.510653260	10.0.0.10	10.0.0.20	TCP	1514 8080 - 39374 [ACK] Seq=228004 Ack=1 Win=500 Len=1448 TStamp=4232957
1188	25.510653512	10.0.0.10	10.0.0.20	TCP	1514 8080 - 39374 [ACK] Seq=220452 Ack=1 Win=500 Len=1448 TStamp=4232957
1189	25.510653577	10.0.0.10	10.0.0.20	TCP	1417 8080 - 39374 [PSH, ACK] Seq=230900 Ack=1 Win=500 Len=1351 TStamp=4232957
1190	25.510671918	10.0.0.20	10.0.0.10	TCP	66 39374 - 8080 [ACK] Seq=1 Ack=229452 Win=501 Len=0 TStamp=1367512729
1191	25.510672084	10.0.0.20	10.0.0.10	TCP	66 39374 - 8080 [ACK] Seq=1 Ack=230900 Win=496 Len=0 TStamp=1367512729
1192	25.510672138	10.0.0.20	10.0.0.10	TCP	66 39374 - 8080 [ACK] Seq=1 Ack=232251 Win=491 Len=0 TStamp=1367512729
1193	25.510672177	10.0.0.10	10.0.0.20	TCP	1514 8080 - 44440 [ACK] Seq=228004 Ack=1 Win=500 Len=1448 TStamp=1175213
1194	25.510672177	10.0.0.10	10.0.0.20	TCP	1514 8080 - 44440 [ACK] Seq=2290452 Ack=1 Win=500 Len=1448 TStamp=1175213
▶ Frame 1187: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth1.0.1, id 0					
Ethernet II, Src: Intel PRO/100 MT [00:0c:29:b4:4d:9f], Dst: Intel PRO/100 MT [00:0c:29:b4:4d:9e] (Intel PRO/100 MT [00:0c:29:b4:4d:9f])					
Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20					
Transmission Control Protocol, Src Port: 8080, Dst Port: 39374, Seq: 228004, Ack: 1, Len: 1448					
Source Port: 8080					
Destination Port: 39374					
[Stream index: 0]					
[TCF Segment Len: 1448]					
Sequence number: 228004 (relative sequence number)					
Sequence number (raw): 4038312195					
[Next sequence number: 229452 (relative sequence number)]					
Acknowledgment number: 1 (relative ack number)					
Acknowledgment number (raw): 1789222458					
1000 = Header Length: 32 bytes (8)					
Flags: 0x10 (ACK)					
Window size: 500					
[Calculate window size: 500]					
[Window size scaling factor: -1 (unknown)]					
Checksum: 0xa0a5 [unverified]					
[Checksum Status: Unverified]					
Urgent pointer: 0					
▶ Options: (12 bytes), NO-Operation (NOP), NO-Operation (NOP), Timestamps					
[SEQ/ACK analysis]					
[timestamps]					
TCP payload (1448 bytes)					
▶ Hypertext Transfer Protocol					

Fig. 8. Tráfego com servidor e Jasmin + Bela + Monstro (3 clientes)

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	0.0000000000	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=1 Ack=1 Win=500 Len=1448 Tsvl=4231435110 T...
2	0.000000376	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=1449 Ack=1 Win=509 Len=1448 Tsvl=423143511...
3	0.000000456	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=2897 Ack=1 Win=509 Len=1448 Tsvl=423143511...
4	0.000000519	10.0.0.10	10.0.0.29	TCP	85	8080 - 39374 [PSH, ACK] Seq=345 Ack=1 Win=509 Len=19 Tsvl=423143...
5	0.0000024066	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=1449 Win=501 Len=0 Tsvl=1366890036 T...
6	0.0000025070	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=2897 Win=496 Len=0 Tsvl=1366890036 T...
7	0.0000025611	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=4341 Win=491 Len=9 Tsvl=1366890036 T...
8	0.0000026205	10.0.0.20	10.0.0.10	TCP	1514	8080 - 39374 [ACK] Seq=1 Ack=4341 Win=491 Len=9 Tsvl=1366890036 T...
9	0.485375494	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=1449 Ack=1 Win=509 Len=1448 Tsvl=423143506...
10	0.485375116	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=50121 Ack=1 Win=509 Len=1448 Tsvl=423143506...
11	0.485375198	10.0.0.10	10.0.0.29	TCP	1330	8080 - 39374 [PSH, ACK] Seq=7260 Ack=1 Win=509 Len=1264 Tsvl=4221...
12	0.485397531	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=5812 Win=501 Len=0 Tsvl=1366891122 T...
13	0.485398522	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=7261 Win=496 Len=0 Tsvl=1366891122 T...
14	0.485399067	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=8524 Win=491 Len=0 Tsvl=1366891122 T...
15	0.749742906	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=8524 Ack=1 Win=509 Len=1448 Tsvl=423143586...
16	0.749742621	10.0.0.10	10.0.0.10	TCP	1514	8080 - 39374 [ACK] Seq=9072 Ack=1 Win=509 Len=1448 Tsvl=423143586...
17	0.749743337	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=14241 Ack=1 Win=509 Len=1448 Tsvl=42314358...
18	0.749743367	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=14241 Ack=1 Win=509 Len=1448 Tsvl=42314358...
19	0.749743458	10.0.0.10	10.0.0.29	TCP	449	8080 - 39374 [PSH, ACK] Seq=14316 Ack=1 Win=509 Len=383 Tsvl=4231...
20	0.749743644	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=9072 Win=501 Len=0 Tsvl=1366891386 T...
21	0.749743930	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=11428 Win=496 Len=0 Tsvl=1366891386 T...
22	0.749743973	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=12868 Win=491 Len=0 Tsvl=1366891386 T...
23	0.749747648	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=14310 Win=485 Len=0 Tsvl=1366891386 T...
24	0.7497471007	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=14699 Win=483 Len=0 Tsvl=1366891386 T...
25	0.9918695822	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=14699 Ack=1 Win=509 Len=1448 Tsvl=423143601...
26	0.9918696027	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=10141 Ack=1 Win=509 Len=1448 Tsvl=423143601...
27	0.9918696153	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=10141 Ack=1 Win=509 Len=1448 Tsvl=423143601...
28	0.991869164	10.0.0.10	10.0.0.29	TCP	434	8080 - 39374 [PSH, ACK] Seq=10843 Ack=1 Win=509 Len=383 Tsvl=4231...
29	0.9918239854	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=16147 Win=501 Len=0 Tsvl=1366891628 T...
30	0.991833843	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=17595 Win=496 Len=0 Tsvl=1366891628 T...
31	0.991831388	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=19043 Win=491 Len=0 Tsvl=1366891628 T...
32	0.991831919	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=19411 Win=489 Len=0 Tsvl=1366891628 T...
33	1.2389029141	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=14011 Ack=1 Win=509 Len=1448 Tsvl=423143603...
36	1.2389029777	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=20859 Ack=1 Win=500 Len=1448 Tsvl=42314363...
37	1.2389029854	10.0.0.10	10.0.0.29	TCP	1514	8080 - 39374 [ACK] Seq=22307 Ack=1 Win=509 Len=1448 Tsvl=42314363...
38	1.2389029820	10.0.0.10	10.0.0.29	TCP	184	8080 - 39374 [PSH, ACK] Seq=23755 Ack=1 Win=509 Len=120 Tsvl=4231...
39	1.2389029485	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=20859 Win=501 Len=0 Tsvl=1366891628 T...
40	1.2389079909	10.0.0.20	10.0.0.10	TCP	66	39374 - 8080 [ACK] Seq=1 Ack=22307 Win=496 Len=0 Tsvl=1366891628 T...

Frame(s) 1000 bytes on wire (1212 bits), 1000 bytes captured (1212 bits) on interface veth0_1, brd 00:00:00:00:00:00

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)

Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20

Transmission Control Protocol, Src Port: 8080, Dst Port: 39374, Seq: 71038, Ack: 1, Len: 1448

Hypertext Transfer Protocol

Fig. 9. Stream 0 para 1 cliente

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

Fig. 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
388	11. 04:18:04.0443	10.0.0.10	10.0.2.21	TCP	434	0888 → 49558 [PSH, ACK] Seq=116096 Ack=1 Win=507 Len=36 Tsv1=332..
389	11. 04:18:07:633	10.0.0.2.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=113208 Win=501 Len=9 Tsv1=4055597532..
390	11. 04:18:09:8285	10.0.0.2.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=114648 Win=496 Len=9 Tsv1=4055597532..
391	11. 04:18:09:8295	10.0.0.2.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=114648 Win=496 Len=9 Tsv1=4055597532..
392	11. 04:18:09:902	10.0.0.2.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=116484 Win=489 Len=9 Tsv1=4055597532..
393	12. 18:06:18:047	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=116464 Ack=1 Win=509 Len=1448 Tsv1=4231633..
394	12. 18:06:18:259	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=117912 Ack=1 Win=509 Len=1448 Tsv1=4231633..
395	12. 18:06:18:338	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=119369 Ack=1 Win=509 Len=1448 Tsv1=4231633..
396	12. 18:06:18:395	10.0.0.10	10.0.0.29	TCP	188	0888 → 39374 [PSH, ACK] Seq=120888 Ack=1 Win=509 Len=128 Tsv1=423..
397	12. 18:06:18:7344	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=117912 Win=501 Len=9 Tsv1=1367088648..
398	12. 18:06:18:8205	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=119368 Win=496 Len=9 Tsv1=1367088648..
399	12. 18:06:18:8635	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=120886 Win=491 Len=9 Tsv1=1367088648..
400	12. 18:06:18:8644	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=120886 Win=491 Len=9 Tsv1=1367088648..
401	12. 18:06:18:8934	10.0.0.20	10.0.0.10	TCP	1514	0888 → 39374 [ACK] Seq=12088644 Ack=1 Win=507 Len=1448 Tsv1=4231633..
402	12. 18:06:18:9574	10.0.0.10	10.0.0.21	TCP	1514	0888 → 49558 [ACK] Seq=117912 Ack=1 Win=507 Len=1448 Tsv1=3223336..
403	12. 18:06:18:95748	10.0.0.10	10.0.0.21	TCP	1514	0888 → 49558 [ACK] Seq=119369 Ack=1 Win=507 Len=1448 Tsv1=3223336..
404	12. 18:06:18:95813	10.0.0.10	10.0.2.21	TCP	186	0888 → 49558 [PSH, ACK] Seq=120888 Ack=1 Win=507 Len=128 Tsv1=332..
405	12. 18:07:17:034	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=117912 Win=501 Len=9 Tsv1=4055597777..
406	12. 18:07:17:562	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=119368 Win=496 Len=9 Tsv1=4055597777..
407	12. 18:07:17:975	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=120886 Win=491 Len=9 Tsv1=4055597777..
408	12. 18:07:18:881	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=120928 Win=491 Len=9 Tsv1=4055597777..
409	12. 18:07:18:9279	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=120928 Ack=1 Win=509 Len=1448 Tsv1=4231633..
410	12. 18:07:18:9308	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=1209308 Ack=1 Win=509 Len=1448 Tsv1=4231633..
411	12. 18:07:18:9914	10.0.0.10	10.0.0.29	TCP	1514	0888 → 39374 [ACK] Seq=1209914 Ack=1 Win=509 Len=1448 Tsv1=4231633..
412	12. 18:07:19:0975	10.0.0.10	10.0.0.29	TCP	148	0888 → 39374 [PSH, ACK] Seq=1209975 Ack=1 Win=509 Len=348 Tsv1=423..
413	12. 18:07:19:1042	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=122376 Win=501 Len=9 Tsv1=1367089153..
414	12. 18:07:19:1099	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=123824 Win=496 Len=9 Tsv1=1367089153..
415	12. 18:07:19:2345	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=125272 Win=491 Len=9 Tsv1=1367089153..
416	12. 18:07:19:2770	10.0.0.20	10.0.0.10	TCP	66	39374 → 8888 [ACK] Seq=1 Ack=125024 Win=489 Len=9 Tsv1=1367089153..
417	12. 18:07:19:2783	10.0.0.10	10.0.2.21	TCP	1514	0888 → 49558 [ACK] Seq=120928 Ack=1 Win=507 Len=1448 Tsv1=3223336..
418	12. 18:07:19:2983	10.0.0.10	10.0.2.21	TCP	1514	0888 → 49558 [ACK] Seq=122378 Ack=1 Win=507 Len=1448 Tsv1=3223336..
419	12. 18:07:19:2987	10.0.0.10	10.0.2.21	TCP	1514	0888 → 49558 [ACK] Seq=123824 Ack=1 Win=507 Len=1448 Tsv1=3223336..
420	12. 18:07:19:3049	10.0.0.10	10.0.2.21	TCP	418	0888 → 49558 [PSH, ACK] Seq=125024 Ack=1 Win=507 Len=352 Tsv1=332..
421	12. 18:07:19:3050	10.0.0.10	10.0.2.21	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=125272 Win=491 Len=9 Tsv1=4055598282..
422	12. 18:07:19:4609	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=123824 Win=496 Len=9 Tsv1=4055598282..
423	12. 18:07:19:5029	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=125272 Win=491 Len=9 Tsv1=4055598282..
424	12. 18:07:19:5443	10.0.0.21	10.0.0.10	TCP	66	49558 → 8888 [ACK] Seq=1 Ack=125024 Win=489 Len=9 Tsv1=4055598282..

Fig. 11. Stream 0 para 2 clientes

No.	Time	Source	Destination	Protocol	Length	Info
9	0.0000934126	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=1 Ack=1 Win=507 Len=1448 Tsv=323323814 T...
10	0.0000934202	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=1 Ack=2897 Win=507 Len=1448 Tsv=323323811 T...
11	0.0000934329	10.0.0.10	10.0.2.21	TCP	65	49558 [PSH, ACK] Seq=1 Ack=2897 Win=507 Len=10 Tsv=323323811 T...
12	0.0000934349	10.0.0.10	10.0.2.21	TCP	66	49558 [ACK] Seq=1 Ack=3435 Win=507 Len=10 Tsv=323323811 T...
13	0.0000934599	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=1449 Win=501 Len=0 Tsv=4055495591 T...
14	0.0000935091	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=2897 Win=496 Len=0 Tsv=4055495591 T...
15	0.0000935487	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=4345 Win=491 Len=0 Tsv=4055495591 T...
16	0.0000935873	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=4364 Win=491 Len=0 Tsv=4055495591 T...
23	0.505223937	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=1 Ack=1 Win=507 Len=1448 Tsv=323323431 T...
24	0.505223314	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=5812 Ack=1 Win=507 Len=1448 Tsv=323323431 T...
25	0.505223315	10.0.0.10	10.0.2.21	TCP	1338	8080 - 49558 [ACK] Seq=5812 Ack=1 Win=507 Len=1448 Tsv=323323431 T...
26	0.5052249900	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=5812 Win=501 Len=0 Tsv=4055495591 T...
27	0.505250407	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=7269 Win=496 Len=0 Tsv=4055495591 T...
28	0.505250819	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=8524 Win=491 Len=0 Tsv=4055495591 T...
39	0.748385329	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=8524 Ack=1 Win=507 Len=1448 Tsv=323323456 T...
40	0.748385464	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=9972 Ack=1 Win=507 Len=1448 Tsv=323323456 T...
41	0.748385533	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=11424 Ack=1 Win=507 Len=1448 Tsv=32332345...
42	0.74838559	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=12806 Ack=1 Win=507 Len=1448 Tsv=32332345...
43	0.748385659	10.0.0.10	10.0.2.21	TCP	449	8080 - 49558 [PSH, ACK] Seq=14316 Ack=1 Win=507 Len=383 Tsv=323323...
44	0.748385660	10.0.0.10	10.0.2.21	TCP	66	49558 - 8080 [ACK] Seq=14316 Ack=1 Win=507 Len=383 Tsv=323323...
45	0.748421535	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=11420 Win=496 Len=0 Tsv=4055495591 T...
46	0.749421949	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=12068 Win=491 Len=0 Tsv=4055495591 T...
47	0.749422343	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=14316 Win=485 Len=0 Tsv=4055495591 T...
48	0.749422735	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=14099 Win=483 Len=0 Tsv=4055495591 T...
57	1.611596339	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=14699 Ack=1 Win=507 Len=1448 Tsv=323323428...
58	1.611596457	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=16147 Ack=1 Win=507 Len=1448 Tsv=323323428...
59	1.611596533	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=17598 Ack=1 Win=507 Len=1448 Tsv=323323428...
60	1.611596592	10.0.0.10	10.0.2.21	TCP	434	8080 - 49558 [PSH, ACK] Seq=19043 Ack=1 Win=507 Len=388 Tsv=323323...
61	1.611596593	10.0.0.10	10.0.2.21	TCP	66	49558 - 8080 [ACK] Seq=19043 Ack=1 Win=507 Len=388 Tsv=323323...
62	1.611596599	10.0.0.10	10.0.2.21	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=17789 Win=496 Len=0 Tsv=4055495591 T...
63	1.611596639	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=10943 Win=491 Len=0 Tsv=4055495591 T...
64	1.611596736	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=19411 Win=489 Len=0 Tsv=4055495591 T...
73	1.255943857	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=19411 Ack=1 Win=507 Len=1448 Tsv=323323258...
74	1.255943981	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=20851 Ack=1 Win=507 Len=1448 Tsv=323323258...
75	1.255944044	10.0.0.10	10.0.2.21	TCP	1514	8080 - 49558 [ACK] Seq=22307 Ack=1 Win=507 Len=1448 Tsv=323323258...
76	1.255944108	10.0.0.10	10.0.2.21	TCP	186	8080 - 49558 [PSH, ACK] Seq=23755 Ack=1 Win=507 Len=120 Tsv=323323...
77	1.255975195	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=20859 Win=501 Len=0 Tsv=4055495591 T...
78	1.255975730	10.0.0.21	10.0.0.10	TCP	66	49558 - 8080 [ACK] Seq=1 Ack=22307 Win=496 Len=0 Tsv=4055495591 T...

Frame 9: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth1_0.1, 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: 10.0.2.21
 Transmission Control Protocol, Src Port: 8080, Dst Port: 49558, Seq: 1, Ack: 1, Len: 1448
 Hypertext Transfer Protocol

Fig. 12. Stream 1 para 2 clientes

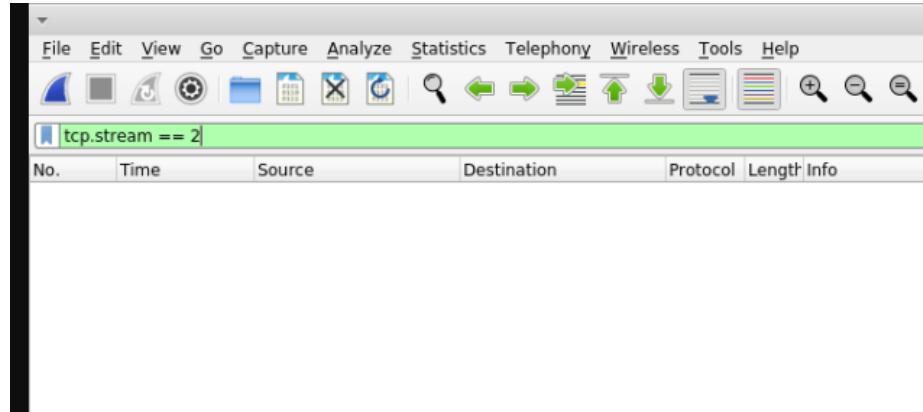


Fig. 13. Stream 2 para 2 clientes

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

Fig. 14. Stream 0 para 3 clientes

Fig. 15. Stream 1 para 3 clientes

Fig. 16. Stream 2 para 3 clientes

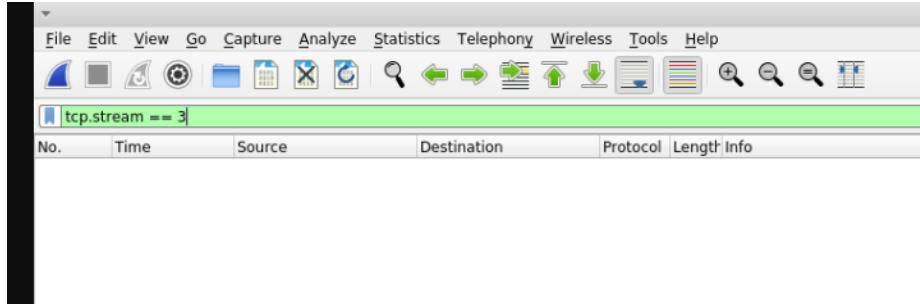


Fig. 17. Stream 3 para 3 clientes

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 seria enviadas para todos os clientes na mesma.

1.2 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 figuras 18, 19 e 20, a largura de banda necessária para visualizar o vídeo *videoB_160_100_200k.mp4* é 66946 bps. Por sua vez, para ver o *videoB_320_200_500k.mp4* é preciso uma de 148018 bps. E por fim, o *videoB_640_400_1000k.mp4* necessita de uma largura de banda de 364370 bps.

```

<?xml version="1.0"?>
<!-- MPD File Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gcd3add+df95d - at 2022-10-13T18:38:59.562Z -->
<MPD xmlns="urn:mpeg:dash:schema:mpd-1.0" profile="urn:mpeg:dash:profile:full-2011">
  <ProgramInformation mediaName="PT0H0M0.500S" type="static" mediaPresentationDuration="PT0H0M0.500S" maxSegmentDuration="PT0H0M0.500S" profiles="urn:mpeg:dash:profile:full-2011"/>
  <VideoItem manifestId="http://gpac-tester.org/test">
    <Representation sourceURL="video_manifest_init.apkt">
      <SegmentList tmscale="15360" duration="7680" start="0" end="7680" indexRange="0-927-970"/>
      <SegmentURL mediaRange="5361-5189" indexRange="5361-5484"/>
      <SegmentURL mediaRange="8188-10266" indexRange="8188-8247"/>
      <SegmentURL mediaRange="12087-14165" indexRange="12087-14165"/>
      <SegmentURL mediaRange="17882-21868" indexRange="17882-17865"/>
      <SegmentURL mediaRange="21869-26924" indexRange="21869-21927"/>
      <SegmentURL mediaRange="26925-31981" indexRange="26925-26927"/>
      <SegmentURL mediaRange="3124-38422" indexRange="3124-35107"/>
      <SegmentURL mediaRange="34345-46917" indexRange="34345-43488"/>
      <SegmentURL mediaRange="46918-53259" indexRange="46918-46917"/>
      <SegmentURL mediaRange="53260-54408" indexRange="53260-53259"/>
      <SegmentURL mediaRange="54408-55819" indexRange="54408-54451"/>
      <SegmentURL mediaRange="55818-55238" indexRange="55818-55057"/>
    </SegmentList>
  </Representation>
</VideoItem>
<Representation sourceURL="audio_manifest_init.apkt">
  <SegmentList tmscale="15360" duration="7680" start="0" end="7680" indexRange="0-927-970"/>
  <SegmentURL mediaRange="5361-5189" indexRange="5361-5484"/>
  <SegmentURL mediaRange="8188-10266" indexRange="8188-8247"/>
  <SegmentURL mediaRange="12087-14165" indexRange="12087-14165"/>
  <SegmentURL mediaRange="17882-21868" indexRange="17882-17865"/>
  <SegmentURL mediaRange="21869-26924" indexRange="21869-21927"/>
  <SegmentURL mediaRange="26925-31981" indexRange="26925-26927"/>
  <SegmentURL mediaRange="3124-38422" indexRange="3124-35107"/>
  <SegmentURL mediaRange="34345-46917" indexRange="34345-43488"/>
  <SegmentURL mediaRange="46918-53259" indexRange="46918-46917"/>
  <SegmentURL mediaRange="53260-54408" indexRange="53260-53259"/>
  <SegmentURL mediaRange="54408-55819" indexRange="54408-54451"/>
  <SegmentURL mediaRange="55818-55238" indexRange="55818-55057"/>
</SegmentList>
</Representation>
</VideoItem>
</ProgramInformation>
</MPD>

```

Fig. 18. Largura de banda para o vídeo de menor qualidade

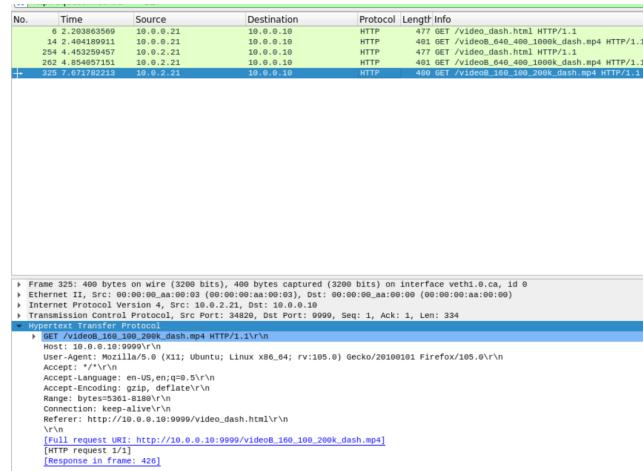
```
<Representation id="2" mimeType="video/mp4" codecs="avc3.640014" width="320" height="200" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="148018">
<BaseURL>videoB_320_200_500k_dash.mp4</BaseURL>
<SegmentList timescale="15360" indexRange="0-10999999999999999">
<SegmentURL mediaRange="0-28_10999999999999999" indexRange="0-928_971"/>
<SegmentURL mediaRange="10999999999999999-17534" indexRange="10999999999999999-17534_11842"/>
<SegmentURL mediaRange="17535-21297" indexRange="17535-17578"/>
<SegmentURL mediaRange="21298-36693" indexRange="21298-21341"/>
<SegmentURL mediaRange="36694-47849" indexRange="36694-36737"/>
<SegmentURL mediaRange="47850-47859" indexRange="47850-47859"/>
<SegmentURL mediaRange="47860-78441" indexRange="47860-47860_78441"/>
<SegmentURL mediaRange="78442-78455" indexRange="78442-78445_78445"/>
<SegmentURL mediaRange="78446-86253" indexRange="78446-78446_86253"/>
<SegmentURL mediaRange="86254-96954" indexRange="86254-86254_96954"/>
<SegmentURL mediaRange="96955-104587" indexRange="96955-96998"/>
<SegmentURL mediaRange="104588-119178" indexRange="104588-104631"/>
<SegmentURL mediaRange="119179-121107" indexRange="119179-119222"/>
<SegmentURL mediaRange="121108-121874" indexRange="121108-121151"/>
<SegmentURL mediaRange="121875-122114" indexRange="121875-121918"/>
</SegmentList>
</Representation>
```

Fig. 19. Largura de banda para o vídeo de intermédia qualidade

```
<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="400" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="364370">
<BaseURL>videoB_640_400_1000k_dash.mp4</BaseURL>
<SegmentList timescale="15360" indexRange="7680">
<SegmentURL mediaRange="22829-40154" indexRange="22829-22872"/>
<SegmentURL mediaRange="40155-49395" indexRange="40155-40198"/>
<SegmentURL mediaRange="49396-87667" indexRange="49396-49349"/>
<SegmentURL mediaRange="87668-116800" indexRange="87668-87711"/>
<SegmentURL mediaRange="116801-143604" indexRange="116801-116944"/>
<SegmentURL mediaRange="143605-143661" indexRange="143605-143647"/>
<SegmentURL mediaRange="192267-212109" indexRange="192267-192310"/>
<SegmentURL mediaRange="212110-237469" indexRange="212110-212153"/>
<SegmentURL mediaRange="237470-259356" indexRange="237470-237513"/>
<SegmentURL mediaRange="259357-259419" indexRange="259357-259400"/>
<SegmentURL mediaRange="294820-299255" indexRange="294820-294863"/>
<SegmentURL mediaRange="299256-300332" indexRange="299256-299299"/>
<SegmentURL mediaRange="300333-300605" indexRange="300333-300376"/>
</SegmentList>
</Representation>
</AdaptationSet>
</Period>
```

Fig. 20. Largura de banda para o vídeo de melhor qualidade

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).

**Fig. 21.** Captura de tramas

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.

No caso do cliente no portátil Bela restringimos o link de modo a que apenas fosse permitida a transmissão do *videoB_160_100_200k.mp4*. Para isso baseámonos na informação disponível no ficheiro ”*video_manifest.mpd*”, já consultada na alínea anterior.

Uma vez que a largura de banda necessária era 66946 bps, restringimos o link do portátil Bela para 95000 bps, como podemos ver pela figura 22.

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

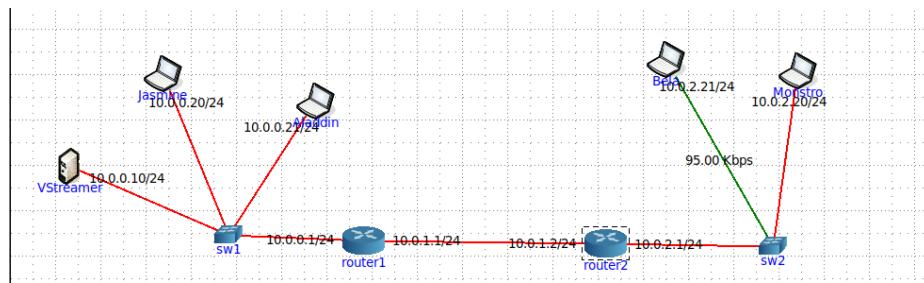


Fig. 22. Topologia com restrição

Após aplicar as restrições necessárias conseguimos ver que no portátil Bela é transmitido o vídeo de menor qualidade e no portátil Alladin o de melhor.

No.	Time	Source	Destination	Protocol	Length	Info
36	32.419837203	10.0.2.21	10.0.0.10	HTTP	381	GET /favicon.ico HTTP/1.1
40	32.425106659	10.0.2.21	10.0.0.10	HTTP	478	GET /video_manifest.mpd HTTP/1.1
51	32.809070574	10.0.2.21	10.0.0.10	HTTP	421	GET /video_manifest_init.mp4 HTTP/1.1
59	32.809070511	10.0.2.21	10.0.0.10	HTTP	404	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
206	49.052046219	10.0.2.21	10.0.0.10	HTTP	402	GET /videoB_320_200_500k_dash.mp4 HTTP/1.1
251	49.051426489	10.0.2.21	10.0.0.10	HTTP	401	GET /videoB_160_100_200k_dash.mp4 HTTP/1.1
373	84.446198006	10.0.0.21	10.0.0.10	HTTP	412	GET / HTTP/1.1
386	84.598234398	10.0.0.21	10.0.0.10	HTTP	366	GET /favicon.ico HTTP/1.1
401	90.512057882	10.0.0.21	10.0.0.10	HTTP	478	GET /video_manifest.mpd HTTP/1.1
409	90.748632684	10.0.0.21	10.0.0.10	HTTP	421	GET /video_manifest_init.mp4 HTTP/1.1
417	99.776538698	10.0.0.21	10.0.0.10	HTTP	401	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1

Fig. 23. Vídeos transmitidos

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

Por 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 o seu bit rate. 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 o rate 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 vídeo buffered e a medida de banda recebida é baixa 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.” (Traduzido do inglês).

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 deste tendo sempre em conta a qualidade de transmissão e de conexão.

1.3 Streaming RTP/RTCP unicast sobre UDP e multicast com anúnicos 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 testados cenários de rede distintos, tendo em conta streaming em unicast que consiste na transmissão apenas entre um sender e um receiver e em multicast que se trata do envio de packets de uma entidade para um grupo de hosts na rede.

Observando apenas o protocol hierarchy das transmissões multicast e unicast efetuadas, conseguimos perceber que a quantidade de dados enviada em cada uma das transmissões é muito semelhante: 139k em unicast e 140k em multicast. No entanto, os 139k são relativos a apenas um cliente enquanto que os 140k são

relativos a 4 clientes. O que demonstra que o serviço multicast é mais eficiente em comparação com o unicast.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	11769	100.0	7902027	152 k	0	0	0
Ethernet	100.0	11769	2.1	164766	3190	0	0	0
Internet Protocol Version 6	0.4	48	0.0	1920	37	0	0	0
User Datagram Protocol	0.0	2	0.0	12	0	0	0	0
Multicast Domain Name System	0.0	2	0.0	90	1	2	90	1
Open Shortest Path First	0.3	41	0.0	1476	28	41	1476	28
Internet Control Message Protocol v6	0.0	5	0.0	80	1	5	80	1
Internet Protocol Version 4	99.3	11683	3.0	233660	4524	0	0	0
User Datagram Protocol	94.2	11092	1.1	88736	1718	0	0	0
Data	93.3	10984	91.8	7253080	140 k	10984	7253080	140 k
Adwin configuration protocol	0.9	108	0.2	11952	231	108	11952	231
Open Shortest Path First	1.7	205	0.1	9020	174	205	9020	174
Internet Control Message Protocol	3.3	386	1.7	136167	2636	386	136167	2636
Address Resolution Protocol	0.3	38	0.0	1064	20	38	1064	20

Fig. 24. Quantidade de dados transmitidos em unicast

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	10131	100.0	7000777	150 k	0	0	0
Ethernet	100.0	10131	2.0	141834	3051	0	0	0
Internet Protocol Version 6	0.1	12	0.0	480	10	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
Internet Control Message Protocol v6	0.1	10	0.0	160	3	10	160	3
Internet Protocol Version 4	99.9	10119	2.9	202380	4354	0	0	0
User Datagram Protocol	99.9	10119	1.2	80952	1741	0	0	0
Session Announcement Protocol	0.7	74	0.3	23076	515	0	0	0
Session Description Protocol	0.7	74	0.3	22200	477	74	22200	477
Real-Time Transport Protocol	97.3	9855	92.7	6488657	139 k	0	0	0
MP4V-E5	97.3	9855	91.0	6370397	137 k	9855	6370397	137 k
Real-time Transport Control Protocol	0.7	73	0.0	2044	43	73	2044	43
Data	1.2	117	0.9	60188	1295	117	60188	1295

Fig. 25. Quantidade de dados transmitidos em multicast

Relativamente ao multicast, em termos de escalabilidade (aumento do número de clientes na rede) é possível concluir que este serviço é mais escalável que o unicast uma vez que responde de forma mais eficiente quando existem vários clientes na rede, o que conduz ao melhor aproveitamento da largura de banda existente e evitando o envio de pacotes desnecessários. Desta forma, em termos de tráfego de rede o serviço multicast também mostra ser mais vantajoso. Também é possível concluir isso observando o wireshark e concluindo que com o serviço multicast se aproveita melhor a largura de banda.

No entanto, o serviço de multicast é bastante mais complexo provocando maior overhead e maiores custos de controlo.

A eficiência do multicast é conseguida com um custo de perda de flexibilidade de serviços em relação ao unicast, que não permite negociar parâmetros de transmissão e de utilização com o emissor. Esta ausência de flexibilidade em multicast pode ser problemática em ambientes de rede heterogéneos.

2 Comentários Finais e Conclusão

Em modo de conclusão, neste trabalho prático aprofundámos o nosso conhecimento relativamente aos vários protocolos de streaming de vídeo e áudio. 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 a partir do protocolo UDP com RTP/RTCP e concluir que no geral o mais benéfico será o multicast.