



Universidade do Minho
Escola de Engenharia

Trabalho Prático 1 (TP1)

Grupo 70

Relatório do Projeto



João Magalhães (PG55956)

Jorge Rodrigues (PG55966)

Rodrigo Gomes (PG56004)

Setembro 2024, Ano Letivo 24/25

Conteúdo

| | | |
|---|------------------|----|
| 1 | Etapa 0 | 5 |
| 2 | Etapa 1 | 8 |
| | 2.1 Questão 1 | 11 |
| 3 | Etapa 2 | 16 |
| | 3.1 Questão 2 | 24 |
| | 3.2 Questão 3 | 26 |
| | 3.3 Questão 4 | 27 |
| | 3.4 Questão 5 | 28 |
| 4 | Etapa 3 | 29 |
| | 4.1 Questão 6 | 31 |
| 5 | Conclusões | 33 |

Lista de Figuras

| | | |
|----|--|----|
| 1 | GUI xeyes | 5 |
| 2 | Gravação do Vídeo A | 6 |
| 3 | Informação e Reprodução do Vídeo A | 6 |
| 4 | Gravação do Vídeo B | 7 |
| 5 | Informação e Reprodução do Vídeo B | 7 |
| 6 | Topologia | 8 |
| 7 | Exemplo de conectividade Servidor - Jasmine | 8 |
| 8 | Exemplo de conectividade Servidor - Bela | 9 |
| 9 | Video em loop nas Jasmine | 9 |
| 10 | Video no portátil Jasmine | 10 |
| 11 | Tráfego capturado no Wireshark | 10 |
| 12 | Execução do Firefox e captura do tráfego | 11 |
| 13 | 1 clientes a reproduzir o video | 11 |
| 14 | 2 clientes a reproduzir o video | 12 |
| 15 | 3 clientes a reproduzir o video | 12 |
| 16 | Taxa em bps necessária | 12 |
| 17 | Pilha protocolar do cliente ffplay | 13 |
| 18 | Pilha protocolar do cliente HTTP | 13 |
| 19 | Gráfico demonstrativo da evolução do débito por número de clientes | 14 |
| 20 | Resultado do comando 1 para a criação da 1º variante do vídeo | 16 |
| 21 | Resultado do comando 2 para a criação da 1º variante do vídeo | 17 |
| 22 | Resultado do comando 3 para a criação da 1º variante do vídeo | 17 |
| 23 | Criação do arquivo MPD | 18 |
| 24 | Ficheiro MPD 1 | 18 |

| | | |
|----|--|----|
| 25 | Ficheiro MPD 2 | 18 |
| 26 | Ficheiro MPD 3 | 18 |
| 27 | Execução dos comandos wget | 19 |
| 28 | Pagina HTML | 19 |
| 29 | Confirmação da comunicação do Servidor-Jasmine | 20 |
| 30 | Confirmação da comunicação do Servidor-Bela | 20 |
| 31 | Execução dos comandos na bash no Servidor | 21 |
| 32 | Visualização da pagina HTML pela Bela | 21 |
| 33 | Execução dos comandos na bash no Alladin | 22 |
| 34 | Visualização da pagina HTML pelo Alladin e pela Bela, respetivamente | 22 |
| 35 | Limitação 1 do link | 23 |
| 36 | Limitação 2 do link | 23 |
| 37 | Captura do Wireshark na limitação 1 | 23 |
| 38 | Captura do Wireshark na limitação 2 | 24 |
| 39 | Débitos para link ilimitado | 24 |
| 40 | Alladin com link ilimitado | 25 |
| 41 | Débito para host Alladin | 25 |
| 42 | Pilha protocolar | 25 |
| 43 | Pilha protocolar ffplay | 26 |
| 44 | Débito medido com ffplay | 26 |
| 45 | Visualização das diferenças na resolução | 27 |
| 46 | Exemplo do protocolo Dash | 28 |
| 47 | Streaming sobre RTP | 29 |
| 48 | Cliente Unicast | 30 |
| 49 | Topologia para Multicast | 30 |
| 50 | Stream Multicast | 31 |
| 51 | Débito com um cliente em RTP | 31 |
| 52 | Débito com 4 clientes em SAP | 32 |

Introdução

Este relatório apresenta uma análise abrangente de várias soluções de *streaming* em tempo real. O projeto visa explorar e avaliar diferentes tecnologias e protocolos de *streaming*, bem como as suas implementações, utilizando o emulador de rede CORE.

O estudo está dividido em três etapas principais: *Streaming* baseado em *HTTP*, *Streaming* adaptativo *MPEG-DASH* e *Streaming* baseado em *UDP* (*Unicast* e *Multicast*).

Ao longo destas etapas, investigamos o desempenho, a eficiência e a escalabilidade de diferentes abordagens de *streaming*.

Os objetivos deste projeto são multifacetados e abrangentes. Primeiramente, buscamos compreender em profundidade as diversas opções disponíveis em termos de pilha de protocolar para aplicações de *streaming*, permitindo-nos avaliar as vantagens e desvantagens de cada abordagem. Paralelamente, pretendemos familiarizar-nos com uma variedade de formatos multimédia e os seus respetivos métodos de empacotamento, essenciais para a eficiente transmissão de conteúdo. Um componente crucial do nosso estudo é ganhar experiência prática significativa com ferramentas de código aberto amplamente utilizadas em implementações de *streaming*, preparando-nos para cenários reais de desenvolvimento. Além disso, propusemo-nos a analisar o desempenho e a escalabilidade de diversas soluções de *streaming*, fornecendo conclusões para a seleção de tecnologias em diferentes contextos. Por fim, objetivamos comparar a eficiência de diferentes protocolos sob várias condições de rede, permitindo-nos compreender como cada solução se comporta em diferentes cenários de conectividade e carga. Através destes objetivos interligados, aspiramos desenvolver uma compreensão integral e prática das tecnologias de *streaming*.

1 Etapa 0

Apesar do enunciado dividir o trabalho prático em 3 fases distintas, optamos por adicionar uma fase que antecede as mesmas. Nesta etapa, o trabalho desenvolvido consiste na configuração do nosso ambiente de modo a conseguirmos realizar as demais tarefas. Para esse efeito, optamos por seguir os passos descritos no enunciado, com o objetivo final de obter dois vídeos básicos feitos através da gravação do ecrã.

As Tarefas 1,2,3 e 4 foram as mais simples. Nestas apenas era pedido que fosse feita a atualização da máquina virtual (Ponto 1), a instalação da ferramenta *FFmpeg* assim como experimentar um pouco com a mesma (Ponto 2 e 3) e finalmente abrimos a aplicação *xeyes* que disponibilizou uma pequena janela gráfica simples, que será alvo das gravações para as próximas tarefas (Ponto 4).



Figura 1: GUI xeyes

As restantes Tarefas, de 5 a 9, correspondem essencialmente à utilização da ferramenta obtida na descrição anterior. Primeiramente, fizemos a nossa primeira captura de ecrã. Para isso utilizamos o comando *ffmpeg ... videoA.mp4*, que deu origem ao *videoA* que será referenciado no decorrer deste trabalho (Ponto 5 e 6). De seguida verificamos a gravação do vídeo. Para isso executamos o comando *ffprobe videoA.mp4* para a ver a informação do ficheiro, e o comando *ffplay videoA.mp4* para reproduzir o vídeo gravado (Ponto 7). Restou apenas repetir o processo, fazendo apenas uma pequena variação no número de *frames* por segundo (aumentamos de 20 *fps* para 30 *fps*), o que pode ser resumido pelo comando que utilizamos *coisa* (Ponto 8 e 9).

```

[XubunCORE_7.5_Large [Em execução] - Oracle VirtualBox
Ficheiro Máquina Ver Entrada Dispositivos Ajuda
Terminal - core@xubuncore: ~
File Edit View Terminal Tabs Help
[1] 22603
core@xubuncore: $ ffplay -f x11grab -video_size 300x226 -framerate 20 -i :0.0+300,200 -pix_fmt yuv420p videoA.mp4
FFmpeg version 4.2.7-Subuntu0.1 Copyright (c) 2000-2022 the FFmpeg developers
built with gcc 9 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
configuration: --prefix=/usr --extra-version='Ubuntu-0.1' --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --enable-gpl --disable-stripping --enable-avresample --disable-filter-resample --enable-avisynth --enable-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-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-libtwink --enable-libvidstab --enable-libvorbis --enable-libwpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-opencl --enable-opengl --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.  0.100 / 58.  0.100
libavfilter     7. 57.100 /  7. 57.100
libavresample   4.  0.  0 /  4.  0.  0
libswscale      5.  5.100 /  5.  5.100
libswresample   5.  0.100 /  5.  0.100
libavfilter     5.  5.100 /  5.  5.100
Input #0, x11grab, from ':0.0+300,200':
  Duration: N/A, start: 1726755966.086665, bitrate: N/A
Stream mapping:
  Stream #0:0 -> rawvideo (native) -> h264 (libx264)
Press [q] to stop, [?] for help
[libx264 @ 0x55ca0d83e0] profile High, level 1.2
[libx264 @ 0x55ca0d83e0] profile High, level 1.2
[libx264 @ 0x55ca0d83e0] config=deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=1 lookahead_threads=1 sliced_threads=0 nr=0 decimate=1 interlace=0 bluray_compat=0 constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=251 keyframe_min=20 intra_refresh=0 r_c=rctf mbtree=1 crf=23.0 qcomp=0.68 qpmin=69 qpstep=4 ip_ratio=1.49 aq=1:1.00
Output #0, rawvideo, to 'videoA.mp4':
Metadata:
  encoder         : Lavf58.29.100
Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuv420p, 300x226, q=-1.1, 20 fps, 10240 tbn, 10240 tbc
Metadata:
  encoder         : Lavc58.54.100 libx264
Side data:
  cpb: maxrate:0 minrate:0 avg:0/0 buffer_size:0 vbv_delay: -1
  frame= 0 kb/s 0.000000 time=00:00:00.000000 bitrate=N/A speed=
  41 fps=33 q=0.0 size= 0Kb time=00:00:00.000000 bitrate=N/A speed=
  0.5kbits/frame= 72 fps=28 q=28.0 size= 0Kb time=00:00:01.40 bitrate=
  0.5kbits/frame= 83 fps=26 q=28.0 size= 0Kb time=00:00:01.95 bitrate=
  0.5kbits/frame= 94 fps=25 q=28.0 size= 0Kb time=00:00:02.50 bitrate=
  0.5kbits/frame= 105 fps=24 q=28.0 size= 0Kb time=00:00:03.50 bitrate=
  0.5kbits/frame= 116 fps=23 q=28.0 size= 0Kb time=00:00:04.00 bitrate=
  0.5kbits/frame= 127 fps=22 q=28.0 size= 0Kb time=00:00:04.50 bitrate=
  0.5kbits/frame= 138 fps=23 q=28.0 size= 0Kb time=00:00:05.00 bitrate=
  0.5kbits/frame= 149 fps=22 q=28.0 size= 0Kb time=00:00:05.50 bitrate=
  0.5kbits/frame= 160 fps=23 q=28.0 size= 0Kb time=00:00:06.00 bitrate=
  0.5kbits/frame= 171 fps=22 q=28.0 size= 0Kb time=00:00:06.50 bitrate=
  0.5kbits/frame= 182 fps=23 q=28.0 size= 0Kb time=00:00:07.00 bitrate=
  0.5kbits/frame= 193 fps=22 q=28.0 size= 0Kb time=00:00:07.50 bitrate=
  0.5kbits/frame= 204 fps=23 q=28.0 size= 0Kb time=00:00:08.00 bitrate=
  0.5kbits/frame= 215 fps=22 q=28.0 size= 0Kb time=00:00:08.50 bitrate=
  0.5kbits/frame= 226 fps=23 q=28.0 size= 0Kb time=00:00:09.00 bitrate=
  0.5kbits/frame= 237 fps=22 q=28.0 size= 0Kb time=00:00:09.50 bitrate=
  0.5kbits/frame= 248 fps=23 q=28.0 size= 0Kb time=00:00:10.00 bitrate=
  0.5kbits/frame= 259 fps=22 q=28.0 size= 0Kb time=00:00:10.50 bitrate=
  0.5kbits/frame= 270 fps=23 q=28.0 size= 0Kb time=00:00:11.00 bitrate=
  0.5kbits/frame= 281 fps=22 q=28.0 size= 0Kb time=00:00:11.50 bitrate=
  0.5kbits/frame= 292 fps=23 q=28.0 size= 0Kb time=00:00:12.00 bitrate=
  0.5kbits/frame= 303 fps=22 q=28.0 size= 0Kb time=00:00:12.50 bitrate=
  0.5kbits/frame= 314 fps=23 q=28.0 size= 0Kb time=00:00:13.00 bitrate=
  0.5kbits/frame= 325 fps=22 q=28.0 size= 0Kb time=00:00:13.50 bitrate=
  0.5kbits/frame= 336 fps=23 q=28.0 size= 0Kb time=00:00:14.00 bitrate=
  0.5kbits/frame= 347 fps=22 q=28.0 size= 0Kb time=00:00:14.50 bitrate=
  0.5kbits/frame= 358 fps=23 q=28.0 size= 0Kb time=00:00:15.00 bitrate=
  0.5kbits/frame= 369 fps=22 q=28.0 size= 0Kb time=00:00:15.50 bitrate=
  0.5kbits/frame= 380 fps=23 q=28.0 size= 0Kb time=00:00:16.00 bitrate=
  0.5kbits/frame= 391 fps=22 q=28.0 size= 0Kb time=00:00:16.50 bitrate=
  0.5kbits/frame= 402 fps=23 q=28.0 size= 0Kb time=00:00:17.00 bitrate=
  0.5kbits/frame= 413 fps=22 q=28.0 size= 0Kb time=00:00:17.50 bitrate=
  0.5kbits/frame= 424 fps=23 q=28.0 size= 0Kb time=00:00:18.00 bitrate=
  0.5kbits/frame= 435 fps=22 q=28.0 size= 0Kb time=00:00:18.50 bitrate=
  0.5kbits/frame= 446 fps=23 q=28.0 size= 0Kb time=00:00:19.00 bitrate=
  0.5kbits/frame= 457 fps=22 q=28.0 size= 0Kb time=00:00:19.50 bitrate=
  0.5kbits/frame= 468 fps=23 q=28.0 size= 0Kb time=00:00:20.00 bitrate=
  0.5kbits/frame= 479 fps=22 q=28.0 size= 0Kb time=00:00:20.50 bitrate=
  0.5kbits/frame= 490 fps=23 q=28.0 size= 0Kb time=00:00:21.00 bitrate=
  0.5kbits/frame= 501 fps=22 q=28.0 size= 0Kb time=00:00:21.50 bitrate=
  0.5kbits/frame= 512 fps=23 q=28.0 size= 0Kb time=00:00:22.00 bitrate=
  0.5kbits/frame= 523 fps=22 q=28.0 size= 0Kb time=00:00:22.50 bitrate=
  0.5kbits/frame= 534 fps=23 q=28.0 size= 0Kb time=00:00:23.00 bitrate=
  0.5kbits/frame= 545 fps=22 q=28.0 size= 0Kb time=00:00:23.50 bitrate=
  0.5kbits/frame= 556 fps=23 q=28.0 size= 0Kb time=00:00:24.00 bitrate=
  0.5kbits/frame= 567 fps=22 q=28.0 size= 0Kb time=00:00:24.50 bitrate=
  0.5kbits/frame= 578 fps=23 q=28.0 size= 0Kb time=00:00:25.00 bitrate=
  0.5kbits/frame= 589 fps=22 q=28.0 size= 0Kb time=00:00:25.50 bitrate=
  0.5kbits/frame= 590 fps=23 q=28.0 size= 0Kb time=00:00:26.00 bitrate=
  0.5kbits/frame= 591 fps=22 q=28.0 size= 0Kb time=00:00:26.50 bitrate=
  0.5kbits/frame= 592 fps=23 q=28.0 size= 0Kb time=00:00:27.00 bitrate=
  0.5kbits/frame= 593 fps=22 q=28.0 size= 0Kb time=00:00:27.50 bitrate=
  0.5kbits/frame= 594 fps=23 q=28.0 size= 0Kb time=00:00:28.00 bitrate=
  0.5kbits/frame= 595 fps=22 q=28.0 size= 0Kb time=00:00:28.50 bitrate=
  0.5kbits/frame= 596 fps=23 q=28.0 size= 0Kb time=00:00:29.00 bitrate=
  0.5kbits/frame= 597 fps=22 q=28.0 size= 0Kb time=00:00:29.50 bitrate=
  0.5kbits/frame= 598 fps=23 q=28.0 size= 0Kb time=00:00:30.00 bitrate=
  0.5kbits/frame= 599 fps=22 q=28.0 size= 0Kb time=00:00:30.50 bitrate=
  0.5kbits/frame= 600 fps=23 q=28.0 size= 0Kb time=00:00:31.00 bitrate=
  0.5kbits/frame= 601 fps=22 q=28.0 size= 0Kb time=00:00:31.50 bitrate=
  0.5kbits/frame= 602 fps=23 q=28.0 size= 0Kb time=00:00:32.00 bitrate=
  0.5kbits/frame= 603 fps=22 q=28.0 size= 0Kb time=00:00:32.50 bitrate=
  0.5kbits/frame= 604 fps=23 q=28.0 size= 0Kb time=00:00:33.00 bitrate=
  0.5kbits/frame= 605 fps=22 q=28.0 size= 0Kb time=00:00:33.50 bitrate=
  0.5kbits/frame= 606 fps=23 q=28.0 size= 0Kb time=00:00:34.00 bitrate=
  0.5kbits/frame= 607 fps=22 q=28.0 size= 0Kb time=00:00:34.50 bitrate=
  0.5kbits/frame= 608 fps=23 q=28.0 size= 0Kb time=00:00:35.00 bitrate=
  0.5kbits/frame= 609 fps=22 q=28.0 size= 0Kb time=00:00:35.50 bitrate=
  0.5kbits/frame= 610 fps=23 q=28.0 size= 0Kb time=00:00:36.00 bitrate=
  0.5kbits/frame= 611 fps=22 q=28.0 size= 0Kb time=00:00:36.50 bitrate=
  0.5kbits/frame= 612 fps=23 q=28.0 size= 0Kb time=00:00:37.00 bitrate=
  0.5kbits/frame= 613 fps=22 q=28.0 size= 0Kb time=00:00:37.50 bitrate=
  0.5kbits/frame= 614 fps=23 q=28.0 size= 0Kb time=00:00:38.00 bitrate=
  0.5kbits/frame= 615 fps=22 q=28.0 size= 0Kb time=00:00:38.50 bitrate=
  0.5kbits/frame= 616 fps=23 q=28.0 size= 0Kb time=00:00:39.00 bitrate=
  0.5kbits/frame= 617 fps=22 q=28.0 size= 0Kb time=00:00:39.50 bitrate=
  0.5kbits/frame= 618 fps=23 q=28.0 size= 0Kb time=00:00:40.00 bitrate=
  0.5kbits/frame= 619 fps=22 q=28.0 size= 0Kb time=00:00:40.50 bitrate=
  0.5kbits/frame= 620 fps=23 q=28.0 size= 0Kb time=00:00:41.00 bitrate=
  0.5kbits/frame= 621 fps=22 q=28.0 size= 0Kb time=00:00:41.50 bitrate=
  0.5kbits/frame= 622 fps=23 q=28.0 size= 0Kb time=00:00:42.00 bitrate=
  0.5kbits/frame= 623 fps=22 q=28.0 size= 0Kb time=00:00:42.50 bitrate=
  0.5kbits/frame= 624 fps=23 q=28.0 size= 0Kb time=00:00:43.00 bitrate=
  0.5kbits/frame= 625 fps=22 q=28.0 size= 0Kb time=00:00:43.50 bitrate=
  0.5kbits/frame= 626 fps=23 q=28.0 size= 0Kb time=00:00:44.00 bitrate=
  0.5kbits/frame= 627 fps=22 q=28.0 size= 0Kb time=00:00:44.50 bitrate=
  0.5kbits/frame= 628 fps=23 q=28.0 size= 0Kb time=00:00:45.00 bitrate=
  0.5kbits/frame= 629 fps=22 q=28.0 size= 0Kb time=00:00:45.50 bitrate=
  0.5kbits/frame= 630 fps=23 q=28.0 size= 0Kb time=00:00:46.00 bitrate=
  0.5kbits/frame= 631 fps=22 q=28.0 size= 0Kb time=00:00:46.50 bitrate=
  0.5kbits/frame= 632 fps=23 q=28.0 size= 0Kb time=00:00:47.00 bitrate=
  0.5kbits/frame= 633 fps=22 q=28.0 size= 0Kb time=00:00:47.50 bitrate=
  0.5kbits/frame= 634 fps=23 q=28.0 size= 0Kb time=00:00:48.00 bitrate=
  0.5kbits/frame= 635 fps=22 q=28.0 size= 0Kb time=00:00:48.50 bitrate=
  0.5kbits/frame= 636 fps=23 q=28.0 size= 0Kb time=00:00:49.00 bitrate=
  0.5kbits/frame= 637 fps=22 q=28.0 size= 0Kb time=00:00:49.50 bitrate=
  0.5kbits/frame= 638 fps=23 q=28.0 size= 0Kb time=00:00:50.00 bitrate=
  0.5kbits/frame= 639 fps=22 q=28.0 size= 0Kb time=00:00:50.50 bitrate=
  0.5kbits/frame= 640 fps=23 q=28.0 size= 0Kb time=00:00:51.00 bitrate=
  0.5kbits/frame= 641 fps=22 q=28.0 size= 0Kb time=00:00:51.50 bitrate=
  0.5kbits/frame= 642 fps=23 q=28.0 size= 0Kb time=00:00:52.00 bitrate=
  0.5kbits/frame= 643 fps=22 q=28.0 size= 0Kb time=00:00:52.50 bitrate=
  0.5kbits/frame= 644 fps=23 q=28.0 size= 0Kb time=00:00:53.00 bitrate=
  0.5kbits/frame= 645 fps=22 q=28.0 size= 0Kb time=00:00:53.50 bitrate=
  0.5kbits/frame= 646 fps=23 q=28.0 size= 0Kb time=00:00:54.00 bitrate=
  0.5kbits/frame= 647 fps=22 q=28.0 size= 0Kb time=00:00:54.50 bitrate=
  0.5kbits/frame= 648 fps=23 q=28.0 size= 0Kb time=00:00:55.00 bitrate=
  0.5kbits/frame= 649 fps=22 q=28.0 size= 0Kb time=00:00:55.50 bitrate=
  0.5kbits/frame= 650 fps=23 q=28.0 size= 0Kb time=00:00:56.00 bitrate=
  0.5kbits/frame= 651 fps=22 q=28.0 size= 0Kb time=00:00:56.50 bitrate=
  0.5kbits/frame= 652 fps=23 q=28.0 size= 0Kb time=00:00:57.00 bitrate=
  0.5kbits/frame= 653 fps=22 q=28.0 size= 0Kb time=00:00:57.50 bitrate=
  0.5kbits/frame= 654 fps=23 q=28.0 size= 0Kb time=00:00:58.00 bitrate=
  0.5kbits/frame= 655 fps=22 q=28.0 size= 0Kb time=00:00:58.50 bitrate=
  0.5kbits/frame= 656 fps=23 q=28.0 size= 0Kb time=00:00:59.00 bitrate=
  0.5kbits/frame= 657 fps=22 q=28.0 size= 0Kb time=00:00:59.50 bitrate=
  0.5kbits/frame= 658 fps=23 q=28.0 size= 0Kb time=00:00:60.00 bitrate=
  0.5kbits/frame= 659 fps=22 q=28.0 size= 0Kb time=00:00:60.50 bitrate=
  0.5kbits/frame= 660 fps=23 q=28.0 size= 0Kb time=00:00:61.00 bitrate=
  0.5kbits/frame= 661 fps=22 q=28.0 size= 0Kb time=00:00:61.50 bitrate=
  0.5kbits/frame= 662 fps=23 q=28.0 size= 0Kb time=00:00:62.00 bitrate=
  0.5kbits/frame= 663 fps=22 q=28.0 size= 0Kb time=00:00:62.50 bitrate=
  0.5kbits/frame= 664 fps=23 q=28.0 size= 0Kb time=00:00:63.00 bitrate=
  0.5kbits/frame= 665 fps=22 q=28.0 size= 0Kb time=00:00:63.50 bitrate=
  0.5kbits/frame= 666 fps=23 q=28.0 size= 0Kb time=00:00:64.00 bitrate=
  0.5kbits/frame= 667 fps=22 q=28.0 size= 0Kb time=00:00:64.50 bitrate=
  0.5kbits/frame= 668 fps=23 q=28.0 size= 0Kb time=00:00:65.00 bitrate=
  0.5kbits/frame= 669 fps=22 q=28.0 size= 0Kb time=00:00:65.50 bitrate=
  0.5kbits/frame= 670 fps=23 q=28.0 size= 0Kb time=00:00:66.00 bitrate=
  0.5kbits/frame= 671 fps=22 q=28.0 size= 0Kb time=00:00:66.50 bitrate=
  0.5kbits/frame= 672 fps=23 q=28.0 size= 0Kb time=00:00:67.00 bitrate=
  0.5kbits/frame= 673 fps=22 q=28.0 size= 0Kb time=00:00:67.50 bitrate=
  0.5kbits/frame= 674 fps=23 q=28.0 size= 0Kb time=00:00:68.00 bitrate=
  0.5kbits/frame= 675 fps=22 q=28.0 size= 0Kb time=00:00:68.50 bitrate=
  0.5kbits/frame= 676 fps=23 q=28.0 size= 0Kb time=00:00:69.00 bitrate=
  0.5kbits/frame= 677 fps=22 q=28.0 size= 0Kb time=00:00:69.50 bitrate=
  0.5kbits/frame= 678 fps=23 q=28.0 size= 0Kb time=00:00:70.00 bitrate=
  0.5kbits/frame= 679 fps=22 q=28.0 size= 0Kb time=00:00:70.50 bitrate=
  0.5kbits/frame= 680 fps=23 q=28.0 size= 0Kb time=00:00:71.00 bitrate=
  0.5kbits/frame= 681 fps=22 q=28.0 size= 0Kb time=00:00:71.50 bitrate=
  0.5kbits/frame= 682 fps=23 q=28.0 size= 0Kb time=00:00:72.00 bitrate=
  0.5kbits/frame= 683 fps=22 q=28.0 size= 0Kb time=00:00:72.50 bitrate=
  0.5kbits/frame= 684 fps=23 q=28.0 size= 0Kb time=00:00:73.00 bitrate=
  0.5kbits/frame= 685 fps=22 q=28.0 size= 0Kb time=00:00:73.50 bitrate=
  0.5kbits/frame= 686 fps=23 q=28.0 size= 0Kb time=00:00:74.00 bitrate=
  0.5kbits/frame= 687 fps=22 q=28.0 size= 0Kb time=00:00:74.50 bitrate=
  0.5kbits/frame= 688 fps=23 q=28.0 size= 0Kb time=00:00:75.00 bitrate=
  0.5kbits/frame= 689 fps=22 q=28.0 size= 0Kb time=00:00:75.50 bitrate=
  0.5kbits/frame= 690 fps=23 q=28.0 size= 0Kb time=00:00:76.00 bitrate=
  0.5kbits/frame= 691 fps=22 q=28.0 size= 0Kb time=00:00:76.50 bitrate=
  0.5kbits/frame= 692 fps=23 q=28.0 size= 0Kb time=00:00:77.00 bitrate=
  0.5kbits/frame= 693 fps=22 q=28.0 size= 0Kb time=00:00:77.50 bitrate=
  0.5kbits/frame= 694 fps=23 q=28.0 size= 0Kb time=00:00:78.00 bitrate=
  0.5kbits/frame= 695 fps=22 q=28.0 size= 0Kb time=00:00:78.50 bitrate=
  0.5kbits/frame= 696 fps=23 q=28.0 size= 0Kb time=00:00:79.00 bitrate=
  0.5kbits/frame= 697 fps=22 q=28.0 size= 0Kb time=00:00:79.50 bitrate=
  0.5kbits/frame= 698 fps=23 q=28.0 size= 0Kb time=00:00:80.00 bitrate=
  0.5kbits/frame= 699 fps=22 q=28.0 size= 0Kb time=00:00:80.50 bitrate=
  0.5kbits/frame= 700 fps=23 q=28.0 size= 0Kb time=00:00:81.00 bitrate=
  0.5kbits/frame= 701 fps=22 q=28.0 size= 0Kb time=00:00:81.50 bitrate=
  0.5kbits/frame= 702 fps=23 q=28.0 size= 0Kb time=00:00:82.00 bitrate=
  0.5kbits/frame= 703 fps=22 q=28.0 size= 0Kb time=00:00:82.50 bitrate=
  0.5kbits/frame= 704 fps=23 q=28.0 size= 0Kb time=00:00:83.00 bitrate=
  0.5kbits/frame= 705 fps=22 q=28.0 size= 0Kb time=00:00:83.50 bitrate=
  0.5kbits/frame= 706 fps=23 q=28.0 size= 0Kb time=00:00:84.00 bitrate=
  0.5kbits/frame= 707 fps=22 q=28.0 size= 0Kb time=00:00:84.50 bitrate=
  0.5kbits/frame= 708 fps=23 q=28.0 size= 0Kb time=00:00:85.00 bitrate=
  0.5kbits/frame= 709 fps=22 q=28.0 size= 0Kb time=00:00:85.50 bitrate=
  0.5kbits/frame= 710 fps=23 q=28.0 size= 0Kb time=00:00:86.00 bitrate=
  0.5kbits/frame= 711 fps=22 q=28.0 size= 0Kb time=00:00:86.50 bitrate=
  0.5kbits/frame= 712 fps=23 q=28.0 size= 0Kb time=00:00:87.00 bitrate=
  0.5kbits/frame= 713 fps=22 q=28.0 size= 0Kb time=00:00:87.50 bitrate=
  0.5kbits/frame= 714 fps=23 q=28.0 size= 0Kb time=00:00:88.00 bitrate=
  0.5kbits/frame= 715 fps=22 q=28.0 size= 0Kb time=00:00:88.50 bitrate=
  0.5kbits/frame= 716 fps=23 q=28.0 size= 0Kb time=00:00:89.00 bitrate=
  0.5kbits/frame= 717 fps=22 q=28.0 size= 0Kb time=00:00:89.50 bitrate=
  0.5kbits/frame= 718 fps=23 q=28.0 size= 0Kb time=00:00:90.00 bitrate=
  0.5kbits/frame= 719 fps=22 q=28.0 size= 0Kb time=00:00:90.50 bitrate=
  0.5kbits/frame= 720 fps=23 q=28.0 size= 0Kb time=00:00:91.00 bitrate=
  0.5kbits/frame= 721 fps=22 q=28.0 size= 0Kb time=00:00:91.50 bitrate=
  0.5kbits/frame= 722 fps=23 q=28.0 size= 0Kb time=00:00:92.00 bitrate=
  0.5kbits/frame= 723 fps=22 q=28.0 size= 0Kb time=00:00:92.50 bitrate=
  0.5kbits/frame= 724 fps=23 q=28.0 size= 0Kb time=00:00:93.00 bitrate=
  0.5kbits/frame= 725 fps=22 q=28.0 size= 0Kb time=00:00:93.50 bitrate=
  0.5kbits/frame= 726 fps=23 q=28.0 size= 0Kb time=00:00:94.00 bitrate=
  0.5kbits/frame= 727 fps=22 q=28.0 size= 0Kb time=00:00:94.50 bitrate=
  0.5kbits/frame= 728 fps=23 q=28.0 size= 0Kb time=00:00:95.00 bitrate=
  0.5kbits/frame= 729 fps=22 q=28.0 size= 0Kb time=00:00:95.50 bitrate=
  0.5kbits/frame= 730 fps=23 q=28.0 size= 0Kb time=00:00:96.00 bitrate=
  0.5kbits/frame= 731 fps=22 q=28.0 size= 0Kb time=00:00:96.50 bitrate=
  0.5kbits/frame= 732 fps=23 q=28.0 size= 0Kb time=00:00:97.00 bitrate=
  0.5kbits/frame= 733 fps=22 q=28.0 size= 0Kb time=00:00:97.50 bitrate=
  0.5kbits/frame= 734 fps=23 q=28.0 size= 0Kb time=00:00:98.00 bitrate=
  0.5kbits/frame= 735 fps=22 q=28.0 size= 0Kb time=00:00:98.50 bitrate=
  0.5kbits/frame= 736 fps=23 q=28.0 size= 0Kb time=00:00:99.00 bitrate=
  0.5kbits/frame= 737 fps=22 q=28.0 size= 0Kb time=00:00:99.50 bitrate=
  0.5kbits/frame= 738 fps=23 q=28.0 size= 0Kb time=00:01:00.00 bitrate=
  0.5kbits/frame= 739 fps=22 q=28.0 size= 0Kb time=00:01:00.50 bitrate=
  0.5kbits/frame= 740 fps=23 q=28.0 size= 0Kb time=00:01:01.00 bitrate=
  0.5kbits/frame= 741 fps=22 q=28.0 size= 0Kb time=00:01:01.50 bitrate=
  0.5kbits/frame= 742 fps=23 q=28.0 size= 0Kb time=00:01:02.00 bitrate=
  0.5kbits/frame= 743 fps=22 q=28.0 size= 0Kb time=00:01:02.50 bitrate=
  0.5kbits/frame= 744 fps=23 q=28.0 size= 0Kb time=00:01:03.00 bitrate=
  0.5kbits/frame= 745 fps=22 q=28.0 size= 0Kb time=00:01:03.50 bitrate=
  0.5kbits/frame= 746 fps=23 q=28.0 size= 0Kb time=00:01:04.00 bitrate=
  0.5kbits/frame= 747 fps=22 q=28.0 size= 0Kb time=00:01:04.50 bitrate=
  0.5kbits/frame= 748 fps=23 q=28.0 size= 0Kb time=00:01:05.00 bitrate=
  0.5kbits/frame= 749 fps=22 q=28.0 size= 0Kb time=00:01:05.50 bitrate=
  0.5kbits/frame= 750 fps=23 q=28.0 size= 0Kb time=00:01:06.00 bitrate=
  0.5kbits/frame= 751 fps=22 q=28.0 size= 0Kb time=00:01:06.50 bitrate=
  0.5kbits/frame= 752 fps=23 q=28.0 size= 0Kb time=00:01:07.00 bitrate=
  0.5kbits/frame= 753 fps=22 q=28.0 size= 0Kb time=00:01:07.50 bitrate=
  0.5kbits/frame= 754 fps=23 q=28.0 size= 0Kb time=00:01:08.00 bitrate=
  0.5kbits/frame= 755 fps=22 q=28.0 size= 0Kb time=00:01:08.50 bitrate=
  0.5kbits/frame= 756 fps=23 q=28.0 size= 0Kb time=00:01:09.00 bitrate=
  0.5kbits/frame= 757 fps=22 q=28.0 size= 0Kb time=00:01:09.50 bitrate=
  0.5kbits/frame= 758 fps=23 q=28.0 size= 0Kb time=00:01:10.00 bitrate=
  0.5kbits/frame= 759 fps=22 q=28.0 size= 0Kb time=00:01:10.50 bitrate=
  0.5kbits/frame= 760 fps=23 q=28.0 size= 0Kb time=00:01:11.00 bitrate=
  0.5kbits/frame= 761 fps=22 q=28.0 size= 0Kb time=00:01:11.50 bitrate=
  0.5kbits/frame= 762 fps=23 q=28.0 size= 0Kb time=00:01:12.00 bitrate=
  0.5kbits/frame= 763 fps=22 q=28.0 size= 0Kb time=00:01:12.50 bitrate=
  0.5kbits/frame= 764 fps=23 q=28.0 size= 0Kb time=00:01:
```

Figura 4: Gravação do Vídeo B

Figura 5: Informação e Reprodução do Vídeo B

2 Etapa 1

Nesta etapa do projeto, o objetivo é explorar e testar o processo de *streaming* de vídeos sobre o protocolo *HTTP*. Apesar de ser uma solução relativamente ineficiente, o *HTTP* continua a ser amplamente utilizado devido à sua preferência na Internet e ao fato de ser o protocolo de aplicação mais popular atualmente.

Ao longo desta fase, configuraremos uma topologia de rede utilizando o emulador *CORE*, composta por um servidor de *streaming*, dispositivos clientes e componentes de rede como *switches* e *routers*. As experiências fornecerão uma visão prática sobre o comportamento do protocolo *HTTP* em aplicações de *streaming* e como este pode ser implementado de maneira funcional, num cenário de transmissão.

As primeiras duas tarefas desta etapa envolvem a configuração do emulador *CORE*. Para isso, criamos uma topologia como está indicado no enunciado e testamos a respetiva conectividade entre os diferentes elementos da topologia (Ponto 1 e 2).

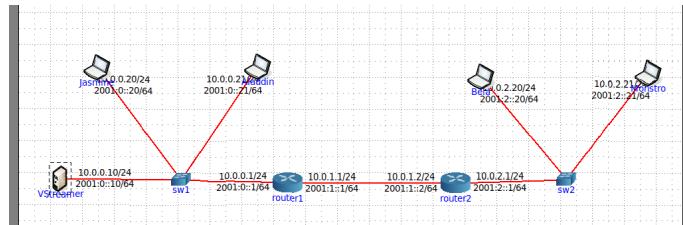


Figura 6: Topologia

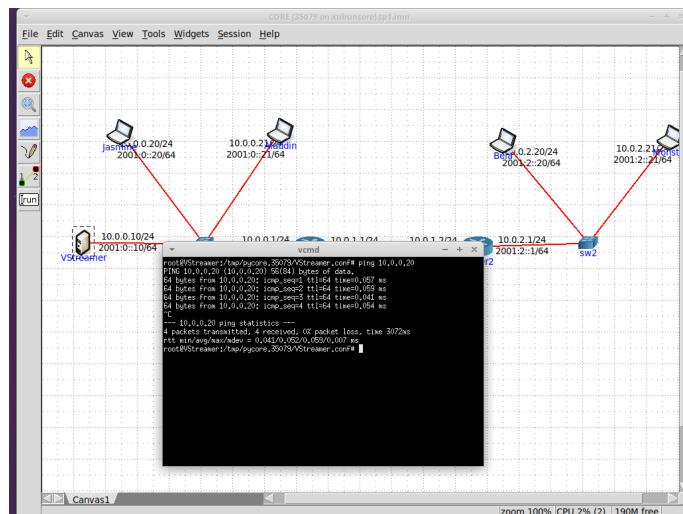


Figura 7: Exemplo de conectividade Servidor - Jasmine

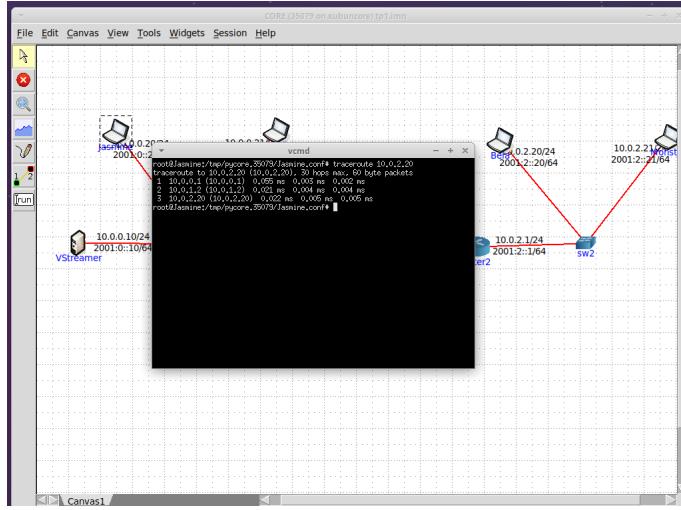


Figura 8: Exemplo de conectividade Servidor - Bela

De seguida, foi nos solicitado que configurássemos o *VLC* no servidor *VStreamer* para realizar o *streaming* por *HTTP* do ficheiro *videoA.mp4*, utilizando o formato de *transcoding Video – Theora + Vorbis (Ogg)*, garantindo que o mesmo seja executado em ciclo continuo (Ponto 3).

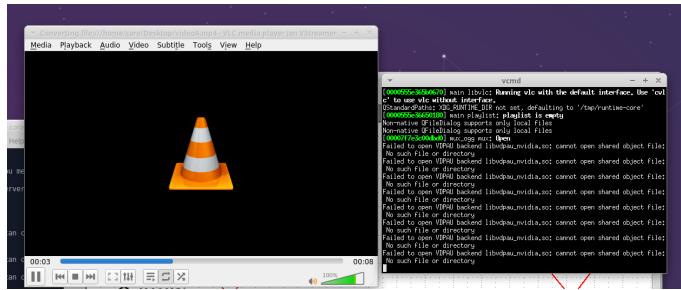


Figura 9: Video em loop nas Jasmine

A próxima tarefa (Ponto 4) pediu que abrissemos um novo terminal no portátil *Jasmine* e configurássemos o *DISPLAY*. Em seguida, iniciamos um segundo *VLC* como cliente, utilizando a opção *Media > Open Network Stream* e inserindo o endereço *http://10.0.0.10:9090/* para acessar o *streaming*.

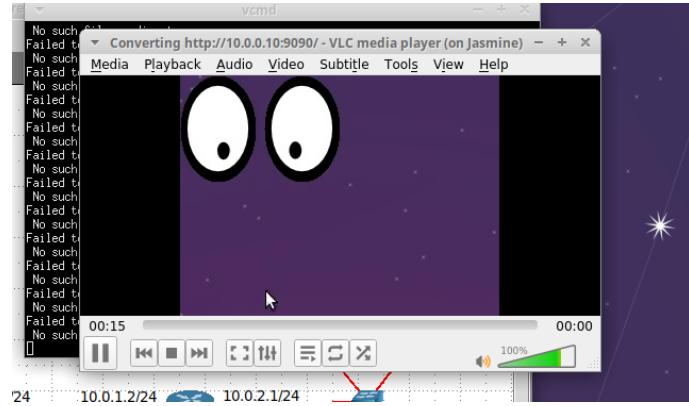


Figura 10: Video no portátil Jasmine

Na tarefa seguinte foi-nos requerido que analisássemos o tráfego através do *Wireshark* na interface de saída do Servidor (Ponto 5).

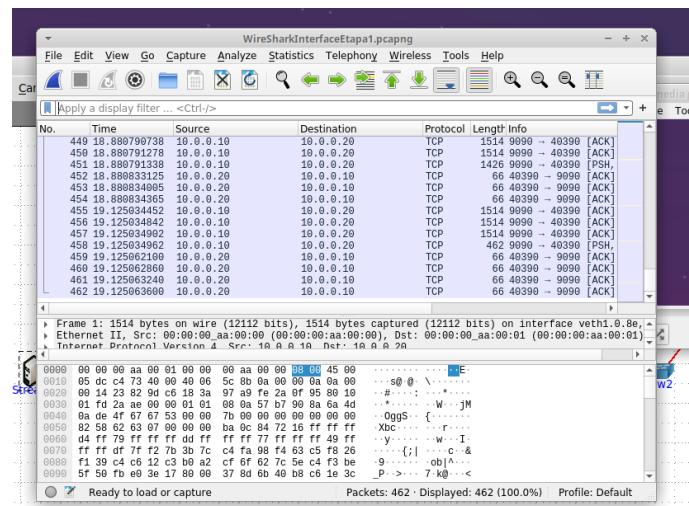


Figura 11: Tráfego capturado no Wireshark

Nas próximas 3 fases criamos uma página *HTML* com o auxilio do código fornecido no guião e dentro da *bash* do portátil Bela, configuramos o *DISPLAY* seguindo os passos descritos e executamos o *Firefox* do *html* criado. Analisamos também o tráfego do mesmo.(Ponto 6, 7 e 8)

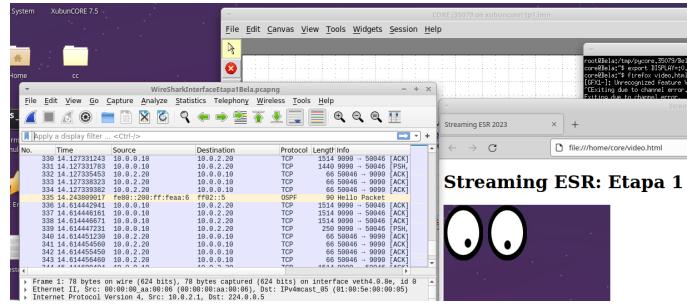


Figura 12: Execução do Firefox e captura do tráfego

Por fim, foi nos solicitado que abrissemos uma novabash para o portátil Monstro, configurando o *DISPLAY* e utilizar o comando *ffplay http://10.0.0.10:9090/* (Ponto 9)

2.1 Questão 1

Os clientes (A) utilizam aproximadamente 130 kbps para se comunicar com o servidor (B), embora, num cenário ideal, o vídeo precise de apenas 20 kbps. Isso indica que o protocolo é bastante ineficiente para a transmissão de vídeo, gerando um consumo de banda muito superior ao necessário. O mesmo pode ser comprovado com as seguintes figuras.

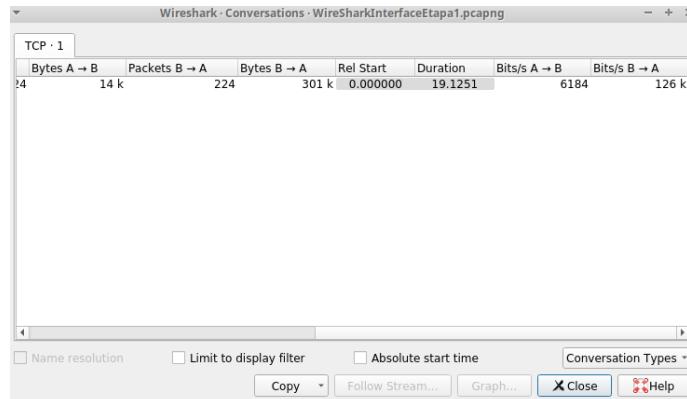


Figura 13: 1 clientes a reproduzir o video

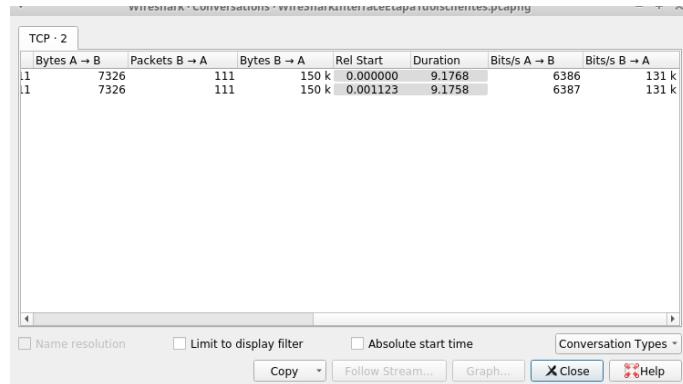


Figura 14: 2 clientes a reproduzir o vídeo

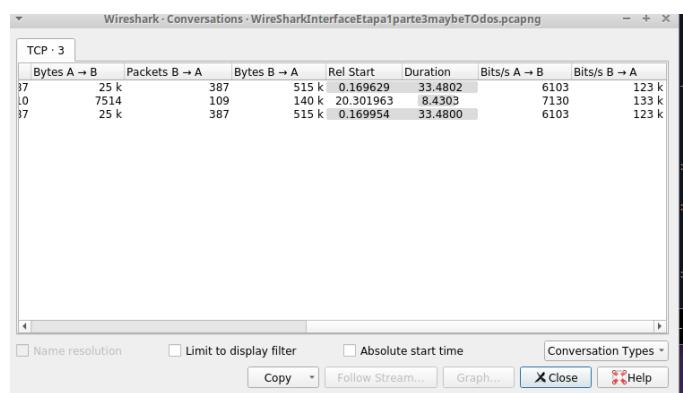


Figura 15: 3 clientes a reproduzir o vídeo

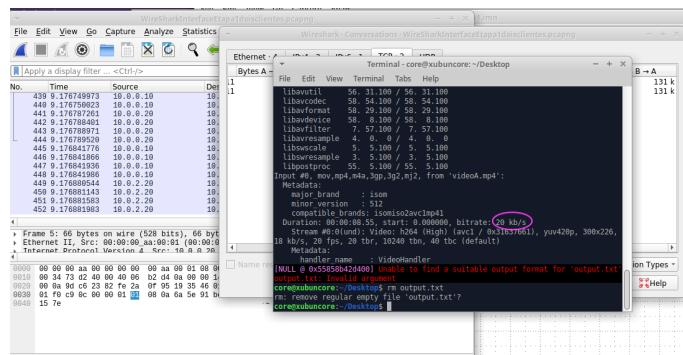


Figura 16: Taxa em bps necessária

ALÍNEA A

No contexto de *streaming* de vídeo via *HTTP*, o protocolo *TCP* desempenha um papel fundamental, pois o *HTTP* opera sobre o *TCP* para garantir a entrega confiável dos dados entre cliente e servidor. No *streaming*, o *TCP* garante que todos os pacotes de dados sejam entregues na íntegra e na ordem correta, o que é essencial

para manter a qualidade do vídeo.

O TCP é orientado à conexão, o que significa que, antes de transferir dados, ele estabelece uma conexão entre as partes através de um processo chamado *Three-Way Handshake*.

Neste protocolo, o vídeo é transmitido em segmentos sequenciais, isto é, os pacotes com os *bits* correspondentes ao segundo x da transmissão nunca serão transmitidos antes dos bits correspondentes ao segundos que antecedem x . Se algum desses segmentos for comprometido, o cliente pode perceber falhas na reprodução do vídeo (*delay*), resultando numa experiência incompleta. Quando não há perdas de dados, o utilizador assiste ao vídeo de forma fluída.

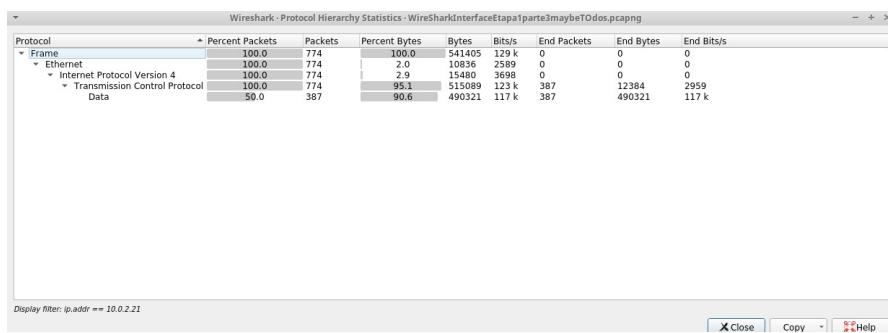


Figura 17: Pilha protocolar do cliente ffplay

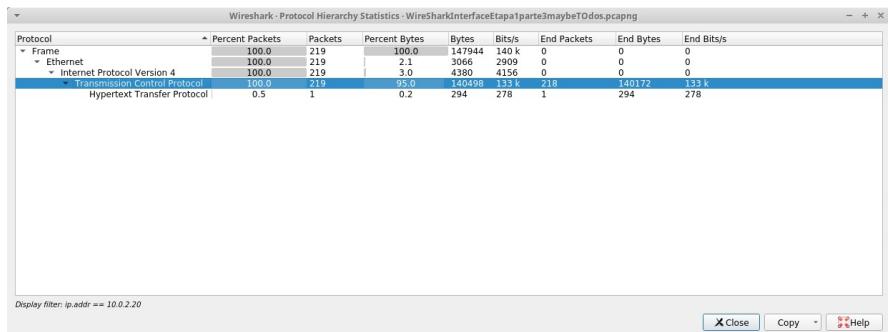


Figura 18: Pilha protocolar do cliente HTTP

Como evidenciado nas figuras 17 e 18

ALÍNEA B

Considerando uma captura com os 3 clientes a receber a transmissão em simultâneo, é evidente a existência de 3 fluxos distintos, um fluxo por cada cliente. Este fenómeno está expresso na figura 15. Apesar dos clientes coexistirem no mesmo intervalo de tempo durante a captura, optamos por elaborar um gráfico cujo o eixo das abscissas corresponde ao número de clientes, e o eixo das ordenadas corresponde ao valor do débito.

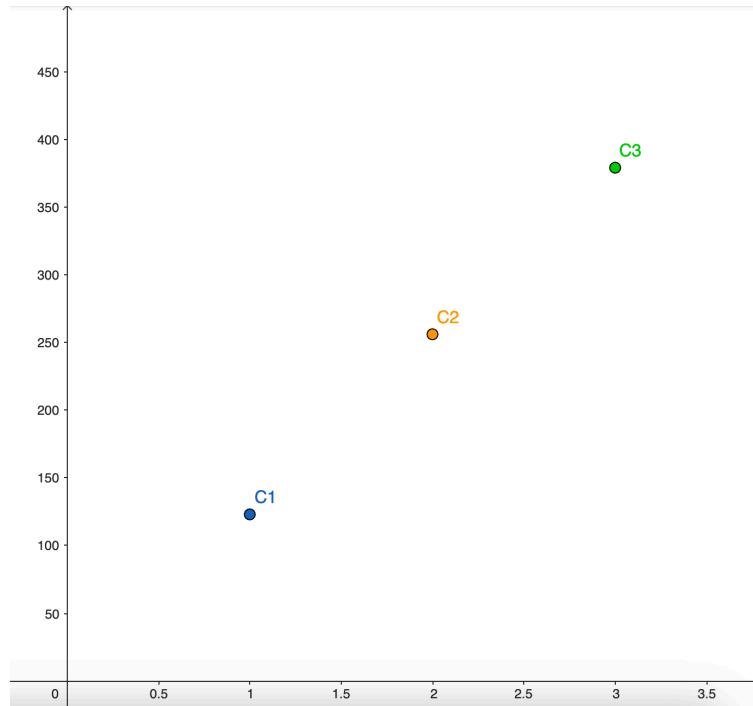


Figura 19: Gráfico demonstrativo da evolução do débito por número de clientes

Os pontos "Cx" ilustradas no gráfico demonstram o débito médio medido para um número "x" de clientes. É importante referir que o débito real não é um valor constante com ilustrado no gráfico. Na realidade, é esperado que haja variações provenientes de falhas na rede ou de outras fontes como hardware. Além disso, se considerássemos que todos os clientes utilizassem a mesma aplicação para receber a *stream*, poderia ser aproximada uma reta linear onde o declive seria o valor do débito.

ALÍNEA C

À primeira vista, a solução não parece uma opção escalável. Isto porque, como referido anteriormente, o débito utilizado para o *streaming* é bastante superior à *bitrate* apresentada no nosso vídeo. Considerando o débito mais elevado medido, neste caso concreto, o utilizador Bela, encontramos um valor de 133 kbps. Optamos por utilizar este valor, pois desta forma conseguimos fazer uma estimativa baseada num cenário menos otimista, que proporciona uma meta mais segura para um possível serviço. Antes de comentar acertadamente a escalabilidade do sistema, é preciso determinar a fórmula matemática que expresse o débito necessário para que o servidor envie um vídeo para N clientes. Desta forma, chegamos à seguinte expressão:

$$D_n = D_c \times N$$

Onde:

- D_n é o débito total necessário para atender N clientes,
- D_c é o débito por cliente (neste caso, 133 **kbps**),
- N é o número de clientes simultâneos.

Para 1000 clientes:

$$D_{1000} = 133 \text{ kbps} \times 1000 = 133 \text{ Mbps}$$

O serviço precisará de suportar 133 *Mbps*, valor que ultrapassa os valores de referência para upload da grande maioria das redes domésticas em Portugal. Se aumentarmos o número de clientes em 10 vezes para 10 000 clientes:

$$D_{10000} = 133 \text{ kbps} \times 10000 = 1.33 \text{ Gbps}$$

O débito necessário passa a 1.33 *Gbps*, novamente um valor demasiado elevado para o serviço em questão.

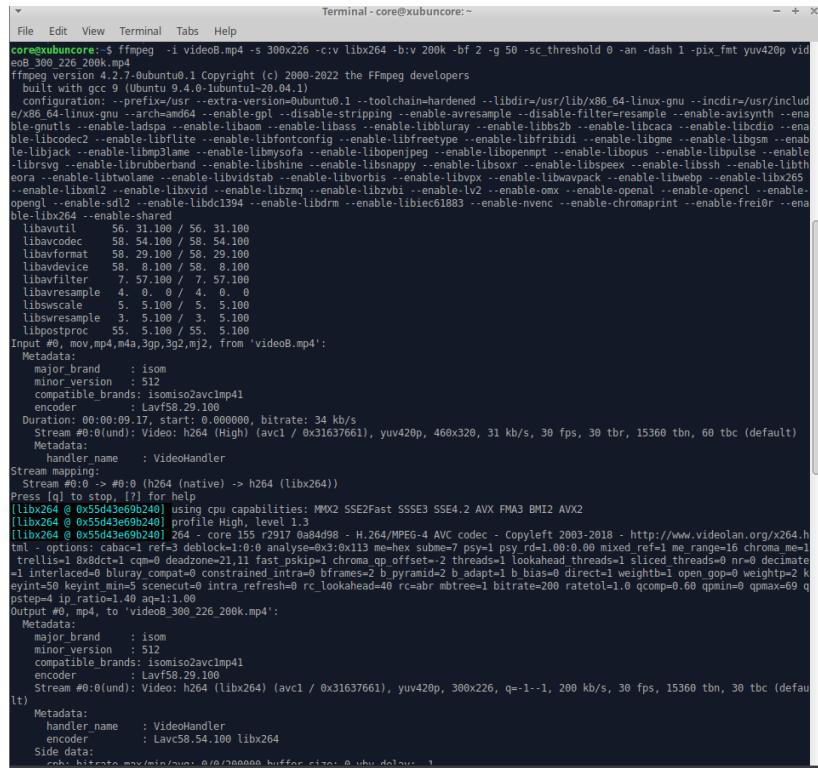
Assim, reparamos que à medida que o número de utilizadores aumenta, a quantidade de dados que o servidor precisa de enviar cresce linearmente, exigindo uma largura de banda significativamente maior. A solução atual não parece ser escalável para cenários de 1000 ou 10 000 utilizadores sem melhorias na capacidade de rede ou sem a adoção de soluções alternativas.

3 Etapa 2

Nesta etapa do projeto, o foco é a criação de um formato *MPEG-DASH* utilizando a ferramenta *ffmpeg*, uma framework de processamento multimédia. A partir do vídeo de entrada, *videoB.mp4*, são geradas três variantes do vídeo em diferentes resoluções. Este processo visa fornecer um serviço de *streaming* com débito adaptativo, permitindo que o sistema ajuste dinamicamente a qualidade do vídeo com base na largura de banda disponível do utilizador.

Além disso, será elaborado um arquivo *XML* que descreve o conteúdo das variantes geradas, conhecido como *Media Presentation Description*, (*MPD*). Esse arquivo é fundamental para a implementação do *streaming* adaptativo, pois fornece informações essenciais sobre as diferentes versões do vídeo e as suas características.

As primeiras três tarefas desta etapa envolvem a geração das três versões do vídeo. Segue se as características dos vídeos.



```
Terminal - core@xubuncore:~
```

```
core@xubuncore:~$ ffmpeg -i videoB.mp4 -s 300x226 -b:v 200k -bf 2 -g 50 -sc_threshold 0 -an -dash 1 -pix_fmt yuv420p via4eot300_226_200k.mp4
ffmpeg version 4.2.7-0ubuntu0.1 Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 9.4.0-lubuntu1-20.04.1
  configuration: --prefix=/usr --extra-version=0ubuntu0.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --enable-amf04 --disable-stripping --enable-avresample --disable-filter=resample --enable-avisynth --enable-gruutis --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgnome --enable-libgsm --enable-libharfbuzz --enable-libimf --enable-libmp3lame --enable-libopenjpeg --enable-libopus --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libvidstab --enable-libvorbis --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libvidstab --enable-libxml2 --enable-libwv --enable-omx --enable-openal --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   5.  5.100 /  5.  5.100
    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 'videoB.mp4':
  Metadata:
    major_brand : isom
    minor_version : 512
    compatible_brands: isomiso2avc1mp41
    encoder : Lavf58.29.100
  Duration: 00:00:09.17, start: 0.000000, bitrate: 34 kb/s
    Stream #0:0[und]: Video: h264 (High) (avc1 / 0x31637661), yuv420p, 460x320, 31 kb/s, 30 fps, 30 tbr, 15360 tbn, 60 tbc (default)
  Metadata:
    handler.name : VideoHandler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x55d43e699240] using cpu capabilities: MMX2 SSE2Fast SSE3 SSE4.2 AVX FMA3 BMI2 AVX2
[libx264 @ 0x55d43e699240] profile High, level 1.3
[libx264 @ 0x55d43e699240] 264 - core 155 r2917 0xa84d98 - H.264/MPEG-4 AVC codec - Copyleft 2003-2018 - http://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x13 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 ipodct=1 cmvc=0 dezone=2,11 fast_pskip=1 chroma_qp_offset=-2 threads=1 lookahead_threads=1 sliced_threads=0 nr=0 decimate=1 interlace=0 constrained_intra=0 bframes=2 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightp=2 K�int=50 point_mine5_smcuct=0 intra_refresh=0 rc_lookahead=48 fcabr mbtree=1 bitrate=200 ratetol=1.0 qcomp=0.60 qpmin=0 qpmax=69 qstep=4 ip_ratio=1.40 an=1.1.00
Output #0, mp4, to 'videoB_300_226_200k.mp4':
  Metadata:
    major_brand : isom
    minor_version : 512
    compatible_brands: isomiso2avc1mp41
    encoder : Lavf58.54.100 libx264
  Side data:
    cbt_bitrate_max/min/avg: 0/0/700000 buffer_size: 0 ubu_dolbyv: 1
```

Figura 20: Resultado do comando 1 para a criação da 1º variante do vídeo

```

core@xubuncore:~$ ffprobe videoB_300_226_200k.mp4
ffprobe version 4.2.7-0ubuntu0.1 Copyright (c) 2007-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-ubuntu1-20.04)
configuration: --prefix=/usr --extra-version='0.1'
  --extra-libs=-lpthread --disable-stripping --enable-avresample --enable-avisynth
  --enable-libaom --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libdca
  --enable-libdct --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm
  --enable-libjack --enable-liblame --enable-libmfx --enable-libopenjpeg --enable-libopus
  --enable-librsvg --enable-librubberband --enable-libssh --enable-libsnappy --enable-libsoxr
  --enable-libspeex --enable-libssh --enable-libtesseract --enable-libtwinkie --enable-libvpx
  --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzvbi --enable-lv2
  --enable-omx --enable-opencore-amr --enable-opencore-amr --enable-opencore-amr
  --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 'videoB_300_226_200k.mp4':
  Metadata:
    major_brand : isom
    minor_version : 512
    compatible_brands: isomiso2avc1mp41
    encoder : Lavf58.29.100
  Duration: 00:00:09.17, start: 0.000000, bitrate: 118 kb/s
  Stream #0:0[und]: Video: h264 (High) (avc1 / 0x31637661), yuv420p, 300x226, 115 kb/s, 30 tbr, 15360 tbn, 60 tbc (default)
    Metadata:
      handler_name : VideoHandler
core@xubuncore:~$ 

```

Figura 21: Resultado do comando 2 para a criação da 1º variante do vídeo

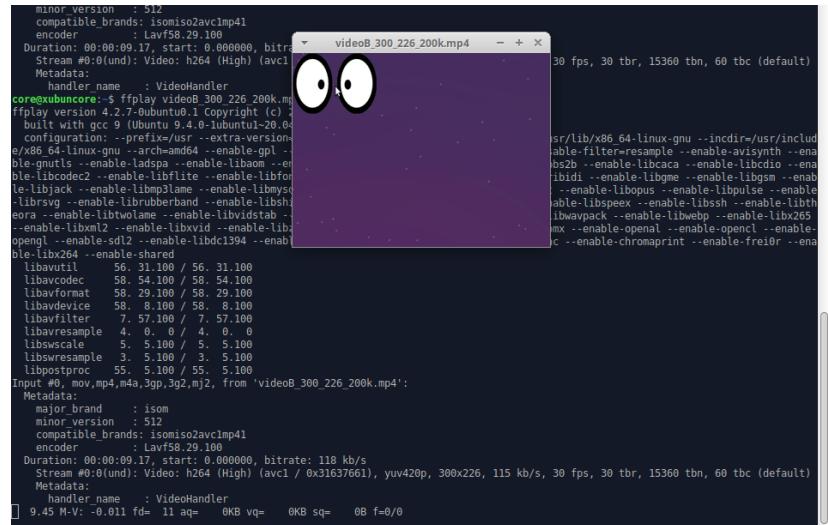


Figura 22: Resultado do comando 3 para a criação da 1º variante do vídeo

Tabela 1: Variantes do vídeo

| Vídeo | Resolução | Taxa de Bits |
|---------------------------|----------------|--------------|
| Vídeo 1 (qualidade baixa) | 300x226 pixels | 118 kbps |
| Vídeo 2 (qualidade média) | 720x540 pixels | 333 kbps |
| Vídeo 3 (qualidade alta) | 960x720 pixels | 573 kbps |

Na tarefa seguinte (ponto 4), ao executar o comando apresentado, o *MP4Box* processará os arquivos de vídeo fornecidos, criados anteriormente, e criará um arquivo *MPD* (*video_manifest*) que contém informações sobre as variantes.

```
core@xubuncore:~$ MP4Box -dash 500 -out video.manifest videoB_300_226_200k.mp4 videoB_720_540_500k.mp4 videoB_960_720_1000k.mp4
[DASH]: Files have non-proportional track layouts (300x226 vs 720x540) but sample size and aspect ratio match, assuming precision issue
[DASH]: Files have non-proportional track layouts (300x226 vs 960x720) but sample size and aspect ratio match, assuming precision issue
DASH-ing files: 0.50s segments 0.50s fragments single sidx per segment
DASHing file videoB_300_226_200k.mp4
DASHing file videoB_720_540_500k.mp4
DASHing file videoB_960_720_1000k.mp4
[DASH] Generating MPD at time 2024-09-26T14:24:20.039Z
core@xubuncore:~$
```

Figura 23: Criação do arquivo MPD

```
-->Representation id="2" mimeType="video/mp4" codecs="avc3.64001P width=720 height=540 frameRate=30 sar=1:1 startWithSAP=0 bandwidth=333098<br-->-->BaseURL><url id="2" 720 540 500k dash.mpd></BaseURL><br-->-->SegmentList timescale="15360" duration="7680" indexRange="928-9717"><br-->-->SegmentURL mediaRange="12129-24049" indexRange="12129-121727"><br-->-->SegmentURL mediaRange="24050-33561" indexRange="24050-240937"><br-->-->SegmentURL mediaRange="35562-45073" indexRange="35562-356137"><br-->-->SegmentURL mediaRange="45950-57653" indexRange="45950-4615137"><br-->-->SegmentURL mediaRange="75656-93563" indexRange="75656-756997"><br-->-->SegmentURL mediaRange="93564-129302" indexRange="93564-936997"><br-->-->SegmentURL mediaRange="129303-143160" indexRange="129303-130917"><br-->-->SegmentURL mediaRange="143160-161502" indexRange="143160-1432937"><br-->-->SegmentURL mediaRange="161503-178154" indexRange="161503-1615467"><br-->-->SegmentURL mediaRange="178155-216253" indexRange="178155-189897"><br-->-->SegmentURL mediaRange="216253-235402" indexRange="216253-2162967"><br-->-->SegmentURL mediaRange="235404-25031" indexRange="235404-2354477"><br-->-->SegmentURL mediaRange="250513-294291" indexRange="250513-2505577"><br-->-->SegmentURL mediaRange="294292-3294357" indexRange="294292-2943357"><br-->-->SegmentURL mediaRange="315722-337337" indexRange="315722-3157657"><br-->-->SegmentURL mediaRange="337352-377096" indexRange="337352-3373957"><br-->-->SegmentURL mediaRange="377097-418127" indexRange="377097-3771277"><br-->-->SegmentURL mediaRange="381059-381674" indexRange="381059-3811927"><br-->-->SegmentList><br-->-->Representation id="3" mimeType="video/mp4" codecs="avc3.64001P width=960 height=720 frameRate=30 sar=1:1 startWithSAP=0 bandwidth=573377<br-->-->BaseURL><url id="3" 960 720 1000 dash.mpd></BaseURL><br-->-->SegmentList timescale="15360" duration="7680" indexRange="928-9717"><br-->-->SegmentURL mediaRange="214114-21410" indexRange="214114-2141077"><br-->-->SegmentURL mediaRange="214114-2141699" indexRange="214114-2145474"><br-->-->SegmentURL mediaRange="21709-36216" indexRange="21709-4174737"><br-->-->SegmentURL mediaRange="36217-4174737" indexRange="36217-4174737"><br-->-->SegmentURL mediaRange="417503-130987" indexRange="417503-13175467"><br-->-->SegmentURL mediaRange="130988-165792" indexRange="130988-1310317"><br-->-->SegmentURL mediaRange="165793-226053" indexRange="165793-1658367">
```

Figura 24: Ficheiro MPD 1

```

<SegmentURL mediaRange="avx3.64001P" indexRange="337352-337353" />
<SegmentURL mediaRange="337352-377096" indexRange="337352-337395" />
<SegmentURL mediaRange="377096-381059" indexRange="377096-377113" />
<SegmentURL mediaRange="381059-381167" indexRange="381059-381102" />
</SegmentList>
<Representation>
<mediaRange>3.3m</mediaRange>
< codecs="avx3.64001P" width="960" height="720" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="573377" />
<BaseURL>video_060_720_1000_dash.mpd</BaseURL>
<SegmentList type="MP3M4" duration="7608" />
<SegmentURL mediaRange="21411-21419" indexRange="21411-21419" />
<SegmentURL mediaRange="21411-21419" indexRange="21411-21454" />
<SegmentURL mediaRange="41700-56281" indexRange="41700-41743" />
<SegmentURL mediaRange="56281-56302" indexRange="56281-56302" />
<SegmentURL mediaRange="107503-130998" indexRange="107503-107546" />
<SegmentURL mediaRange="130998-165759" indexRange="130998-131031" />
<SegmentURL mediaRange="165793-226693" indexRange="165793-165836" />
<SegmentURL mediaRange="226693-251468" indexRange="226693-227979" />
<SegmentURL mediaRange="251468-284304" indexRange="251468-251511" />
<SegmentURL mediaRange="284305-313967" indexRange="284305-284348" />
<SegmentURL mediaRange="313968-377623" indexRange="313968-313972" />
<SegmentURL mediaRange="377623-441083" indexRange="377623-377664" />
<SegmentURL mediaRange="412655-412698" indexRange="412655-412698" />
<SegmentURL mediaRange="441084-511045" indexRange="441084-441127" />
<SegmentURL mediaRange="511046-546751" indexRange="511046-546751" />
<SegmentURL mediaRange="546751-588377" indexRange="546751-546794" />
<SegmentURL mediaRange="588378-649280" indexRange="588378-588421" />
<SegmentURL mediaRange="649281-656274" indexRange="649281-649324" />
<SegmentURL mediaRange="656275-656994" indexRange="656275-656994" />
</SegmentList>
</Representation>
</MediaSelectionSet>
</Period>
</MPD>

```

Figura 25: Ficheiro MPD 2

Figura 26: Ficheiro MPD 3

Na sequência, foi solicitado que importássemos dois *scripts* essenciais para a continuação do guião, utilizando o

comando `wget`. Após completar com sucesso esta etapa, criamos uma página *HTML* intitulada *video_dash.html*, que incorpora as informações fornecidas na tarefa (ponto 5 e 6).

```
core@xubuncore:~$ wget http://cdn.dashjs.org/latest/dash.all.min.js
--2024-09-26 15:25:59-- http://cdn.dashjs.org/latest/dash.all.min.js
Resolving cdn.dashjs.org (cdn.dashjs.org)... 194.210.238.72, 194.210.238.75, 2001:600:c01:1::c2d2:ee48, ...
Connecting to cdn.dashjs.org (cdn.dashjs.org)|194.210.238.72|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-javascript]
Saving to: 'dash.all.min.js'

dash.all.min.js          [=====] 775,25K   188KB/s  in 4,1s

2024-09-26 15:26:05 (188 KB/s) - 'dash.all.min.js' saved [793860]

core@xubuncore:~$ wget http://cdn.dashjs.org/latest/dash.all.debug.js
--2024-09-26 15:26:13-- http://cdn.dashjs.org/latest/dash.all.debug.js
Resolving cdn.dashjs.org (cdn.dashjs.org)... 194.210.238.72, 194.210.238.75, 2001:600:c01:1::c2d2:ee48, ...
Connecting to cdn.dashjs.org (cdn.dashjs.org)|194.210.238.72|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-javascript]
Saving to: 'dash.all.debug.js'

dash.all.debug.js         [=====] 2,92M   131KB/s  in 21s

2024-09-26 15:26:34 (144 KB/s) - 'dash.all.debug.js' saved [3062440]

core@xubuncore:~$
```

Figura 27: Execução dos comandos `wget`

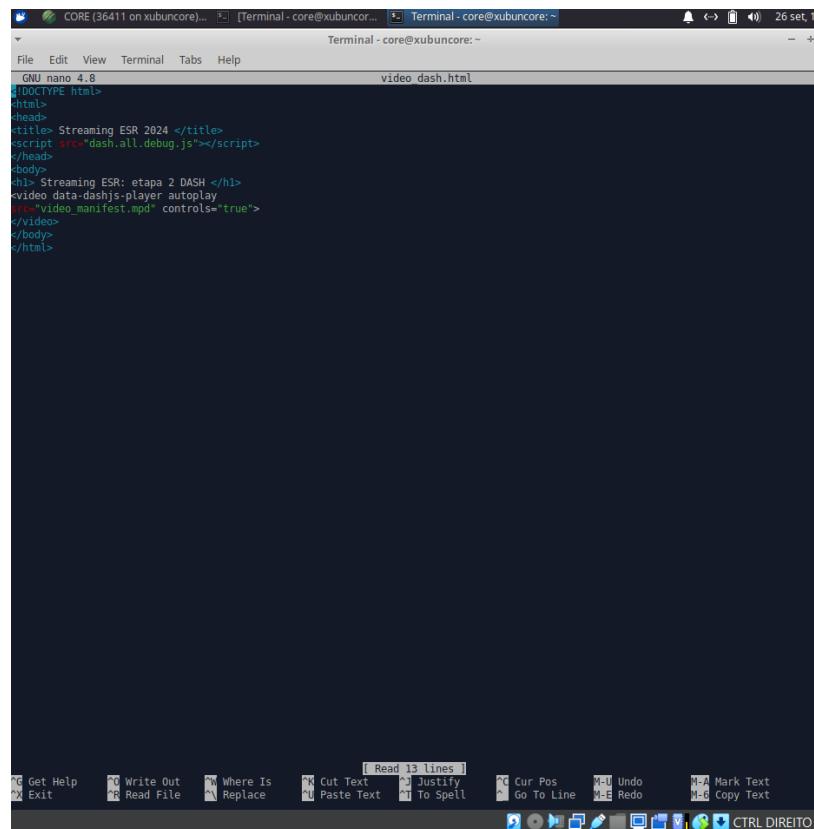


Figura 28: Pagina HTML

Para confirmar que toda a topologia está a funcionar adequadamente, realizamos um novo teste de conectividade. Como demonstrado nas imagens seguintes, todos os componentes da rede estavam a operar conforme o esperado, sem apresentar falhas de comunicação. (ponto 7)

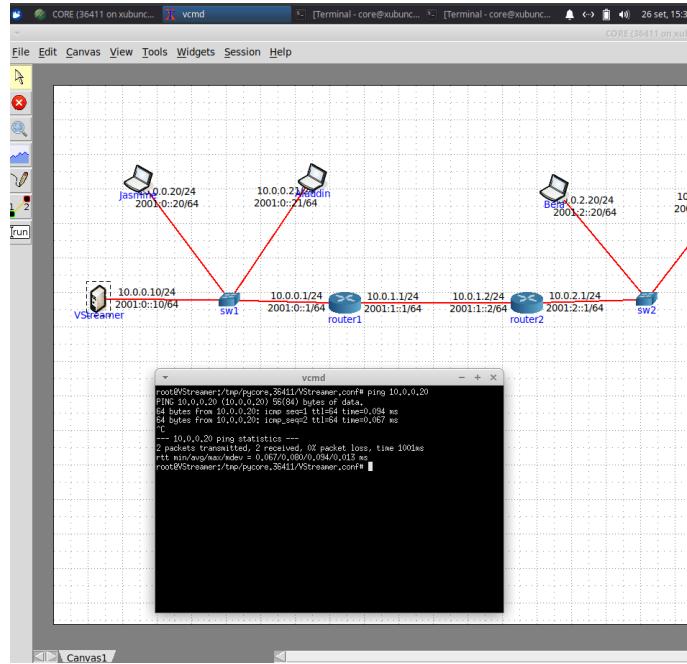


Figura 29: Confirmação da comunicação do Servidor-Jasmine

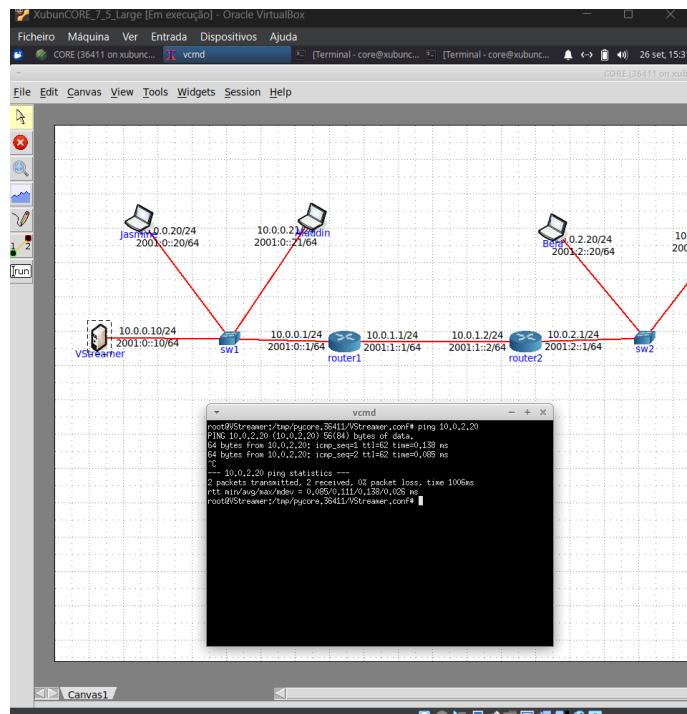


Figura 30: Confirmação da comunicação do Servidor-Bela

De seguida, acedemos ao servidor *VStreamer* e executamos o comando disponibilizado para que o conteúdo possa ser acessado por meio de um navegador ou cliente *HTTP*, utilizando o endereço <http://10.0.0.10:8888>. Por

outras palavras, o VStreamer atua como uma ponte entre a infraestrutura de armazenamento e os utilizadores finais, permitindo a distribuição eficiente de conteúdo multimédia.

Para testar a visualização do conteúdo, utilizamos o portátil Bela com o navegador *Firefox*. (pontos 8 e 9)

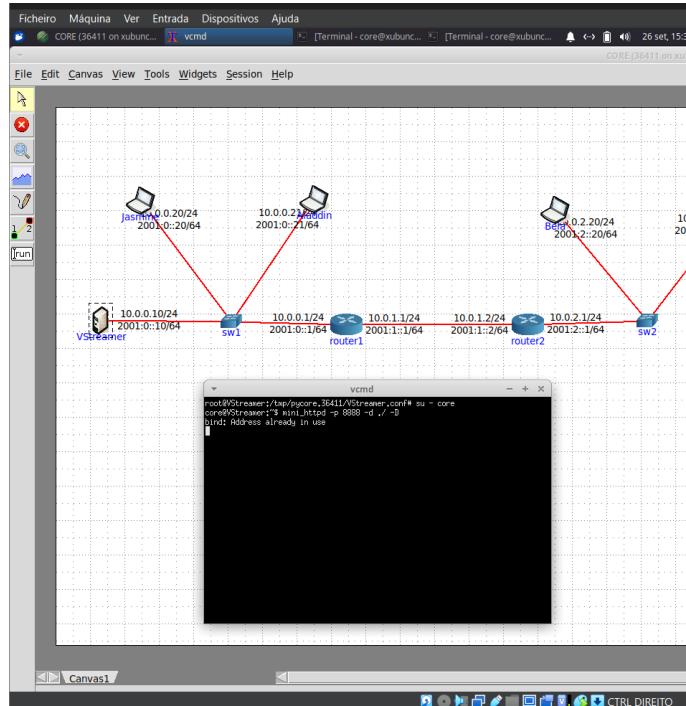


Figura 31: Execução dos comandos na bash no Servidor

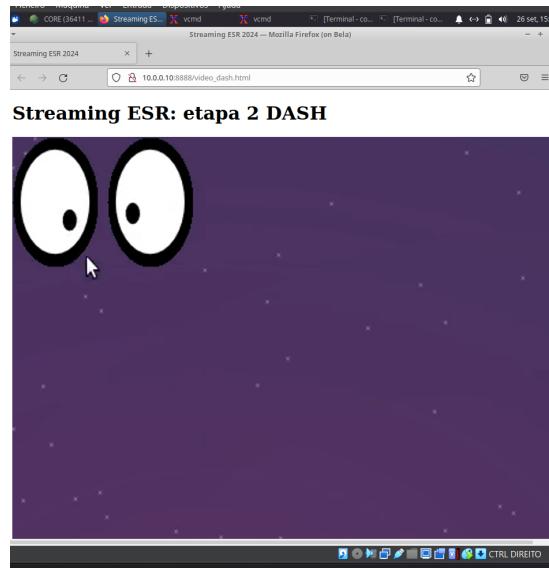


Figura 32: Visualização da pagina HTML pela Bela

Uma vez que já havia uma instância do *Firefox* ativa, não foi possível iniciar uma nova no portátil Alladin,

pois os *hosts* partilham o mesmo sistema de ficheiros. Para contornar esse problema, criamos um novo perfil, conforme descrito no guião, e assim foi possível visualizar o conteúdo com o portátil Alladin. (ponto 10)

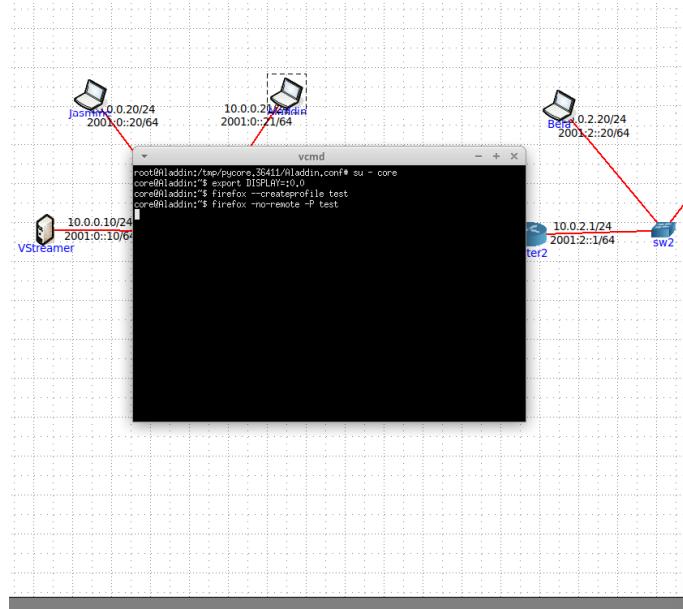


Figura 33: Execução dos comandos na bash no Alladin

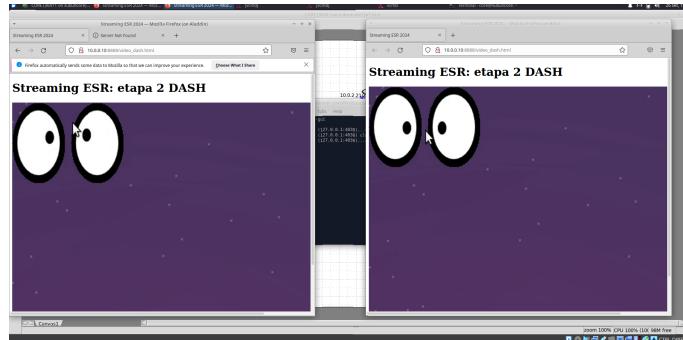


Figura 34: Visualização da pagina HTML pelo Alladin e pela Bela, respetivamente

À medida que tentávamos visualizar o conteúdo da página *HTML*, capturamos o tráfego de rede utilizando o *Wireshark*. (ponto 11)

Por fim, para terminar as etapas, foi pedido que diminuíssemos a dimensão do vídeo transmitido no portátil Bela. Para tal, na topologia ajustamos a capacidade do link (*switch - BELA*) o que limitou a largura de banda disponível e obrigou o sistema a selecionar o vídeo de menor dimensão. (ponto 12)

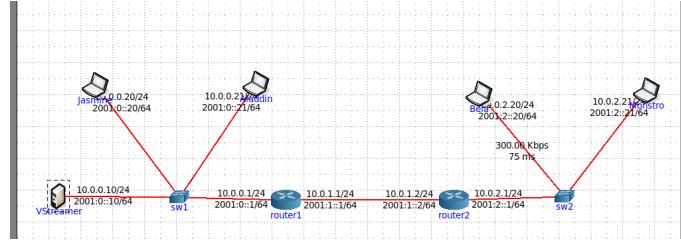


Figura 35: Limitação 1 do link

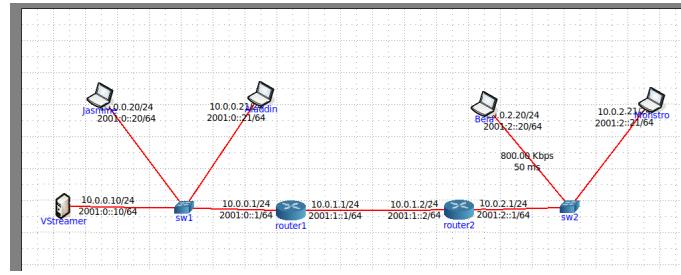


Figura 36: Limitação 2 do link

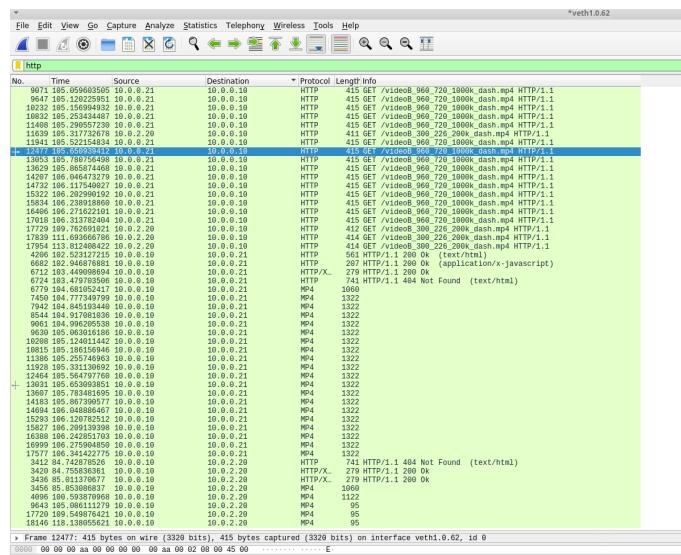


Figura 37: Captura do Wireshark na limitação 1

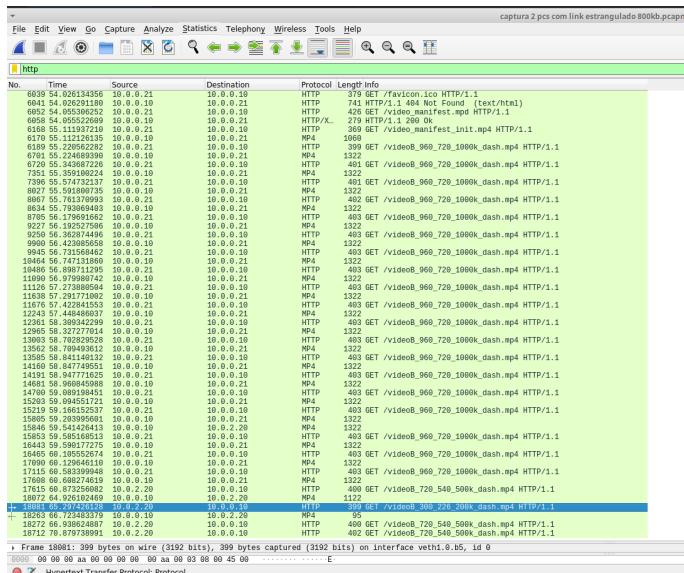


Figura 38: Captura do Wireshark na limitação 2

3.1 Questão 2

Partindo da captura feita sobre a topologia representada na figura 35, apenas com o *host* Alladin, procuramos determinar a largura de banda necessária para que o cliente consiga receber o vídeo no *Firefox*.

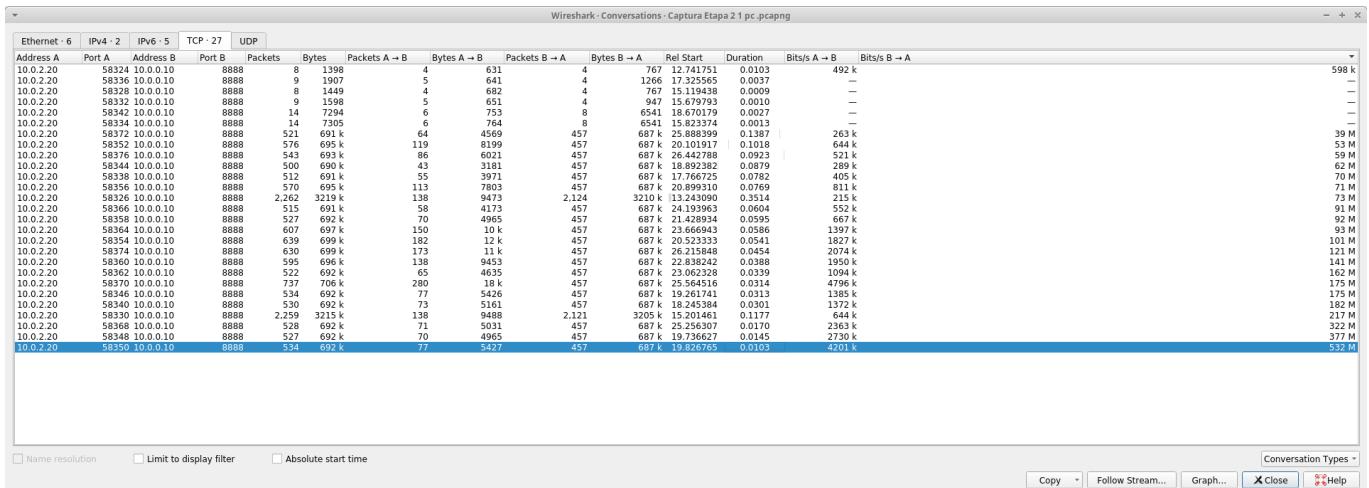


Figura 39: Débitos para link ilimitado

Dos resultados, deparamo-nos com uma particularidade do *Dash*. Quando o link é ilimitado, encontramos um pico no valor do débito que, certamente, ultrapassa o valor necessário para que ocorra a transmissão. Desta forma, optamos por uma solução que limite o débito na rede, mas que garanta que o cliente obtém o vídeo de maior qualidade. Observando o ficheiro *MPD*, sabemos que links com capacidade inferior a 573 kbps (expresso

na figura 25) não serão capazes de ver o vídeo de maior resolução. Assim, gradualmente, fomos subindo a banda larga até encontrar o valor mínimo para qual o cliente recebe o vídeo de maior resolução. No nosso caso, o limite de 800 kbps foi suficiente para que o Alladin, no *Firefox*, conseguisse carregar o vídeo de maior qualidade.

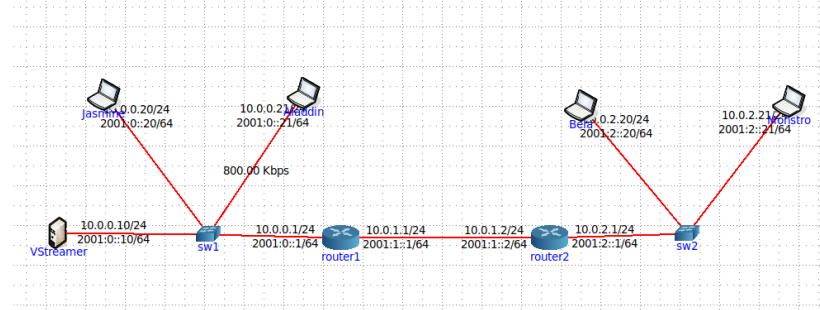


Figura 40: Alladin com link ilimitado

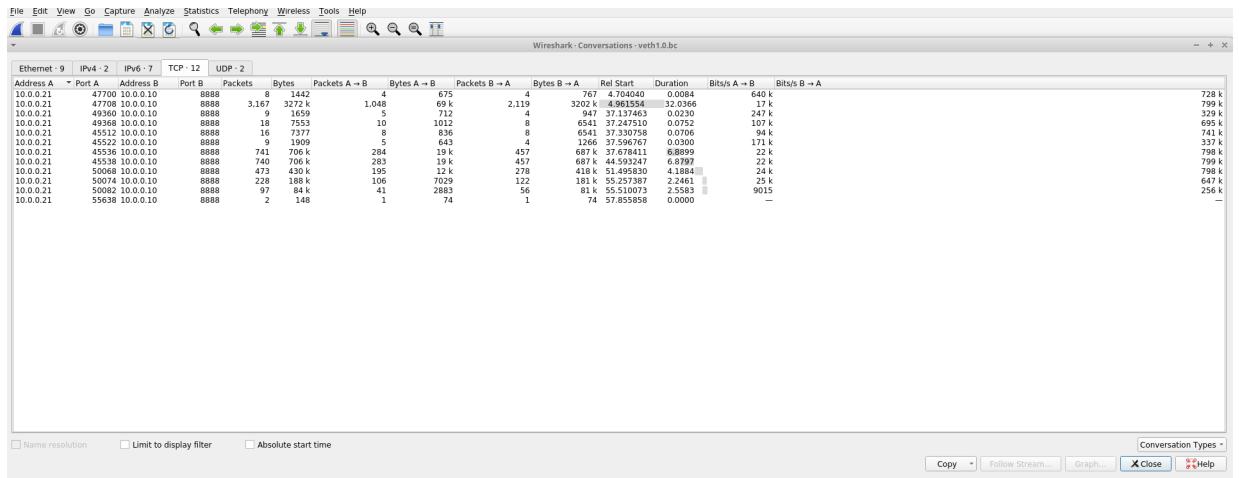


Figura 41: Débito para host Alladin

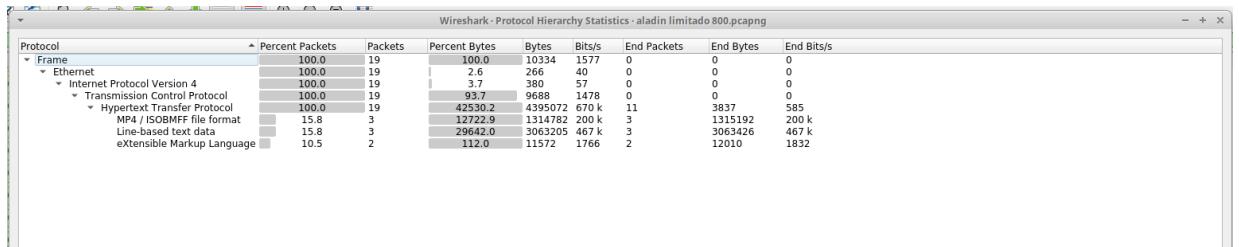


Figura 42: Pilha protocolar

Com o uso do *Wireshark*, como exposto na figura 41, vemos que o débito para o cliente Alladin é de 799 kbps, sendo este o mínimo medido pelo grupo para que o cliente consiga observar o vídeo de maior resolução.

Finalmente, podemos observar a pilha protocolar da transmissão. Observando a figura 42, como esperado, vemos que o *DASH* funciona sobre *HTTP*, o que implica o uso de *TCP* e todo o *overhead* da criação de sessões deste protocolo. Maior parte dos dados transmitidos são referentes ao formato *MP4* e também aos respetivos dados sob a forma de texto.

3.2 Questão 3

A utilização de clientes diferentes deve, à partida, requisitar valores de largura de banda distintos. Na questão anterior, na figura 42, observamos como é a pilha protocolar para um cliente que utilize o *Firefox*. No caso de utilizar o *ffplay*, obtemos a seguinte pilha:

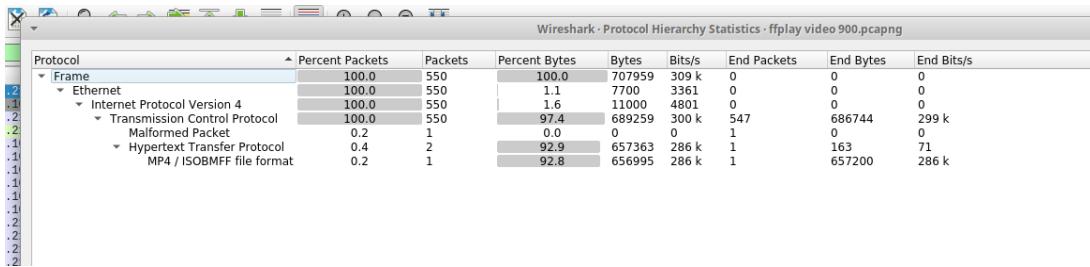


Figura 43: Pilha protocolar *ffplay*

Rapidamente percebemos que, a pilha da figura 43 é bem mais minimalista do ponto de vista protocolar, do que a explorada anteriormente. Desta forma, apesar de não ser a mais bem preparada para funcionar na *web*, é bastante mais leve no que toca a *overhead* dos protocolos.

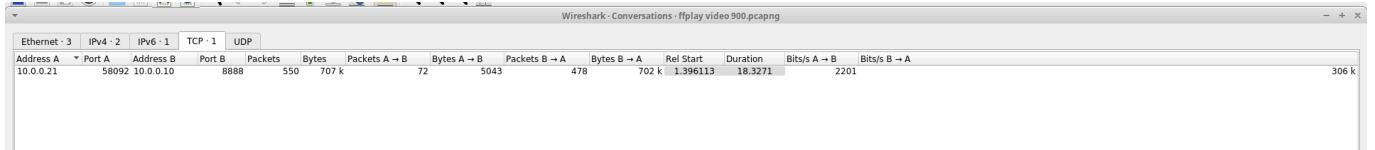


Figura 44: Débito medido com *ffplay*

Comparando os débitos da figura 41 com os da figura 44, corroboramos as conclusões tiradas só com a observação da pilha protocolar. Desta forma, conseguimos afirmar que o cliente *Firefox*, para visualizar o mesmo vídeo, precisou de uma largura de banda muito superior à do cliente *ffplay*.

Assim, concluímos que o cliente *Firefox* é bastante mais pesado em termos de largura de banda do que o cliente *ffplay*. Um dos motivos é a questão da pilha protocolar mais severa face ao *ffplay*, mas é importante referir que o serviço *DASH* foi desenvolvido para a transmissão *web* e possui propriedades como a adaptabilidade à ligação, algo que o *ffplay* não suporta. Deste modo, o *ffplay* é de facto mais leve, mas não oferece as condições que o outro serviço proporciona.

3.3 Questão 4

Começando pela análise do ficheiro *MPD*, gerado pelo uso do *MP4Box*, podemos verificar, com base no tamanho gerado, qual é a largura de banda necessária para receber o respetivo *chunk*.

Nas figuras 24,25 e 26, percebemos que uma largura de banda entre 121 e 333 *kbytes per second* seria ideal para que a Bela obtivesse a resolução mais baixa. Este fenómeno dá-se devido ao funcionamento do *DASH*, onde o vídeo transmitido é ajustado à largura de banda disponível. No caso do Alladin, se não houver qualquer tipo de restrição na largura de banda, é esperado que o vídeo seja o de maior resolução.

Passando para a topologia, o grupo utilizou duas configurações distintas na 35 e 36

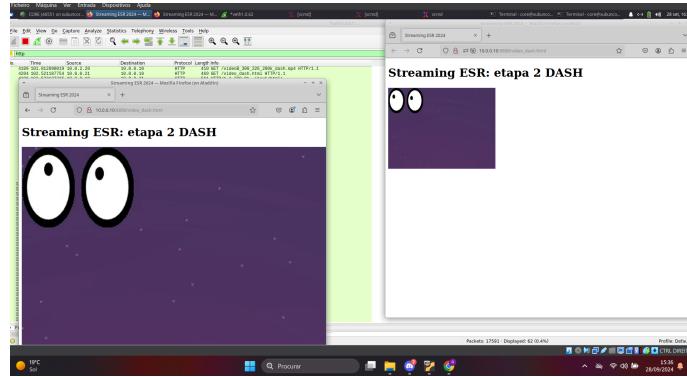


Figura 45: Visualização das diferenças na resolução

Na figura 37, vemos que a Bela tem um limite de 300 *kbytes per second*, enquanto que na segunda, o limite é de 800 *kbytes per second* (teoricamente suficiente para receber o vídeo de melhor resolução). No primeiro caso, conseguimos observar que a bela pede imediatamente ao servidor o vídeo de pior resolução, como era expectável. No entanto, numa segunda abordagem 38, embora fosse de esperar que a Bela pedisse o vídeo de melhor resolução de forma constante, esta acaba por transitar para o meio termo, por complicações de protocolo, congestão da rede ou até *delay* no link de transferência de dados.

Na figura 37, é observável os dois tipos de pedidos feitos pelos clientes, cada um correspondente à respetiva resolução. No caso do Aladin, encontramos pedidos para o vídeo de maior resolução, visto que não há qualquer limitação de banda do seu lado. Já no caso da Bela, observamos que os pedidos são diretamente feitos para o vídeo de menor resolução, assim como esperado. A figura 45 retrata o efeito observado pelos dois utilizadores. Finalmente, a segunda captura é um caso interessante onde o cliente Bela ajustou gradualmente a resolução do vídeo. Este fenómeno é capturado no *Wireshark* pelos diferentes pedidos *HTTP* feitos com resoluções diferentes.

3.4 Questão 5

O *DASH* é um protocolo de *streaming* adaptativo que divide o conteúdo multimédia em pequenos segmentos (*chunks*), permitindo que o cliente ajuste dinamicamente a qualidade do vídeo com base nas condições de rede. O funcionamento do *DASH* envolve a criação de um ficheiro *MPD*, que contém informações sobre as diferentes versões de qualidade do vídeo (*bitrate*, resolução, etc.) e a forma como os segmentos são organizados. Este ficheiro *MPD* é importante no sentido em que permite que o cliente escolha a melhor qualidade de reprodução com base na largura de banda disponível e nas capacidades do dispositivo. Segue-se uma imagem representativa de como as condições do vídeo podem ser adaptadas dependendo das condições da rede.

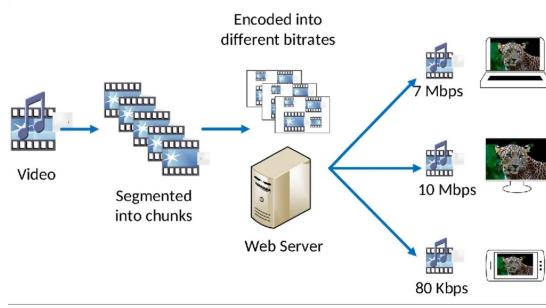


Figura 46: Exemplo do protocolo Dash

No caso explorado na Etapa 2, partimos de um vídeo inicial, o *videoB*, onde começamos por criar as 3 versões com resoluções crescentes. Assim, o servidor que executa o *DASH*, primeiramente informa o cliente sobre o ficheiro *MPD* que, por sua vez, faz o pedido do bloco respetivo, tendo em conta a sua banda larga. Assim, encontramos cenários como os das figuras 37 e 36.

Na questão 1, o modelo utilizado foi o de *streaming* direto via *HTTP*, onde cada cliente (*VLC*, *Firefox*, *ffplay*) recebe o fluxo completo de vídeo diretamente do servidor, sem a possibilidade de adaptar a qualidade do mesmo de acordo com a largura de banda. Este modelo gera uma maior carga no servidor e na rede já que, o mesmo vídeo, é transmitido integralmente para todos os clientes, independentemente das condições da rede ou da capacidade do cliente de reproduzi-lo em melhor qualidade.

Resumidamente, o *DASH* oferece uma solução mais eficiente e escalável em comparação ao modelo de *streaming* utilizado na questão 1, adaptando-se às condições de rede e reduzindo a sobrecarga no servidor. Assim, em cenários com muitos utilizadores, como 1000 ou 10000 clientes, para *streaming* direto sobre *HTTP*, o débito aumenta linearmente com o número de clientes, enquanto no *DASH* a sobrecarga pode ser equilibrada.

4 Etapa 3

Para a etapa final, vamos visualizar e experimentar com protocolos que funcionam sobre *sockets UDP*, um funcionamento diferente dos demais explorados neste relatório. Para esse efeito, em *Unicast*, vamos utilizar o protocolo *RTP/RTCP*, e, para *Multicast*, utilizaremos o protocolo *SAP*. Neste último caso, foi preciso criar uma nova topologia.

As primeiras 2 tarefas desta etapa são bastante semelhantes ao que tem sido feito anteriormente. De qualquer forma, iniciamos o *CORE* e testamos a respetiva conectividade (passo 1 e 2). De seguida, iniciamos uma sessão no servidor sobre o protocolo *RTP* com destino no cliente *Monstro* (passo 3).

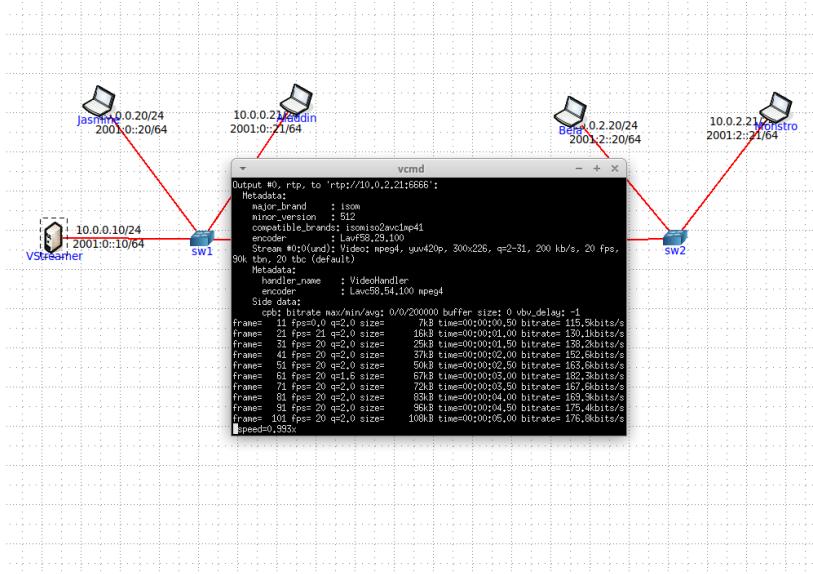


Figura 47: Streaming sobre RTP

Já resta apenas ativar um cliente *ffplay* e capturar o tráfego à saída do servidor, semelhante às tarefas da etapa 1 (passos 3,4 e 5).

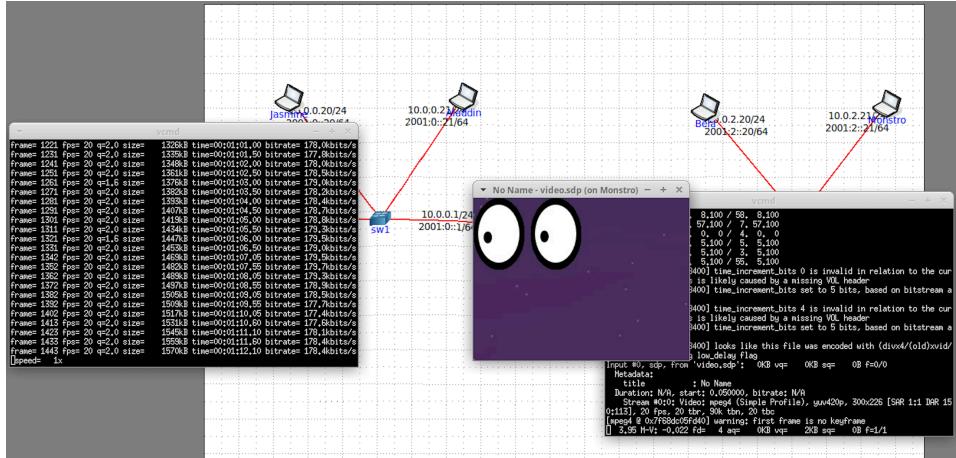


Figura 48: Cliente Unicast

De seguida, avançamos para o *streaming Multicast*. Primeiramente, desenvolvemos uma nova topologia como indicado no guião, e testamos a respetiva conectividade (passos 6 e 7).

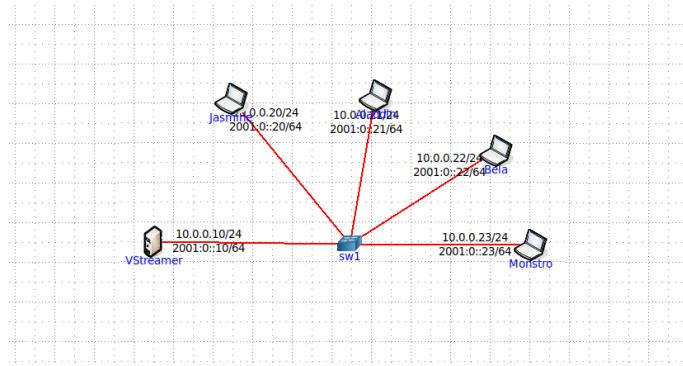


Figura 49: Topologia para Multicast

Para terminar, correspondem à inicialização da transmissão no host *VStreamer*, assim como os respetivos clientes nos restantes *hosts* da topologia. Para esse efeito, começamos pela sessão *SAP* no lado do servidor, e a escuta da mesma através do *ffplay* no lado dos clientes (passos 8,9,10).

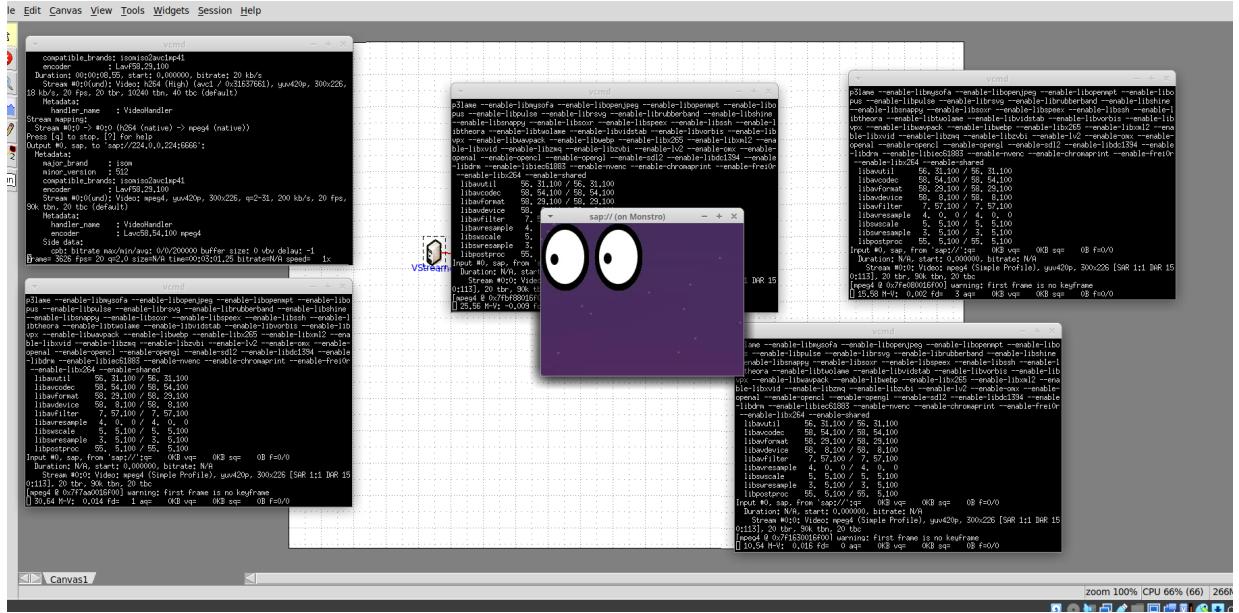


Figura 50: Stream Multicast

4.1 Questão 6

No que toca a *streaming* de conteúdo para vários clientes, existem duas vertentes distintas, mais concretamente, *Unicast* e *Multicast*. O *live streaming* em *Unicast* foi o mais explorado neste relatório. Para efeitos de comparação, vamos optar por observar dados recolhidos para *streaming* através do protocolo *RTP*, pois, como o protocolo usado na vertente *Multicast*, funciona sobre *UDP*. O *RTP/RTCP* é uma das opções mais viáveis para *streaming*, onde se foca principalmente na experiência do cliente, onde algumas perdas de pacotes são toleráveis.

Para *Multicast*, foi utilizado o protocolo *SAP*, um protocolo que permite a um servidor transmitir para múltiplos clientes interessados num determinado conteúdo. Este protocolo define um determinado endereço *IP* único, no qual todos os clientes interessados recebem os datagramas com esse endereço como destino. Assim, o servidor transmite uma única vez para N clientes que se encontram numa determinada rede.

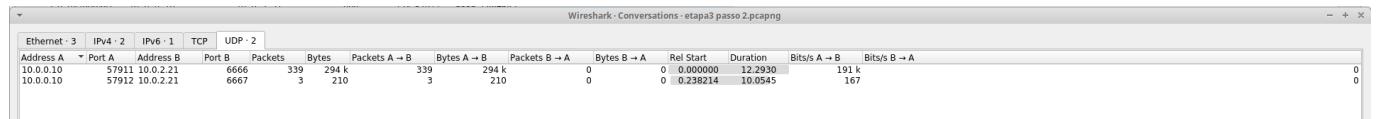


Figura 51: Débito com um cliente em RTP

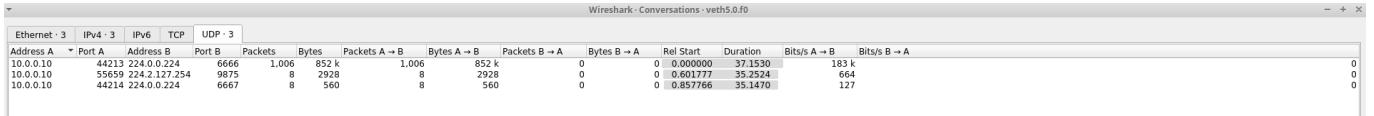


Figura 52: Débito com 4 clientes em SAP

O *Unicast* é a solução menos escalável entre as duas referidas. No que toca a escalabilidade em número de clientes, como visto na figura 51 e na Questão 1, a largura de banda necessária para exercer as funções de servidor é linear no número de clientes. Relativamente ao tráfego na rede, a questão é novamente linear, na medida em que os pacotes são replicados a nível do servidor por cada cliente, o que gera uma congestão elevada na rede do servidor.

Já o *Multicast* é a opção mais escalável. Como pode ser observado na figura 45, existe apenas um endereço para o qual o nosso servidor envia dados. Contudo, estão 4 clientes a serem servidos com um só débito. Deste modo, teoricamente, o número de clientes é irrelevante, pois existe um débito constante para o número de clientes. A congestão também é aliviada, pois as réplicas não ficam na responsabilidade do servidor, mas sim nas camadas de rede.

Observando cenários com 1000 clientes, em *Unicast*, devemos utilizar a fórmula explorada na questão 1. Assim, retiramos que o débito necessário para a essa transmissão será de 191 *Mbps*, ($191 \text{ kbps} * 1000 \text{ clientes}$). Para 10000 clientes, será preciso 1.91 *Gbps*, ($191 \text{ kbps} * 10000 \text{ clientes}$). Avançando para *Multicast*, para 1000 clientes, encontramos um débito de 183 kbps, que se mantém para os 10000 clientes. Desta forma, fica evidente a eficiência do *Multicast* face ao *Unicast*. Contudo, é preciso alertar para a dificuldade que é gerir uma rede *Multicast*. Apesar do débito manter-se, existe algum tráfego extra necessário para manter uma rede *Multicast*. No entanto, contabilizando o tráfego de controlo, *Multicast* continua a ser a opção mais escalável.

5 Conclusões

Este trabalho prático permitiu uma exploração aprofundada de várias soluções de *streaming* em tempo real, utilizando o emulador CORE para simular diferentes topologias de rede. Ao longo das três etapas do projeto, foram analisadas e comparadas diferentes abordagens de *streaming*, cada uma com suas características, vantagens e limitações.

Na primeira etapa, focada no *streaming* via *HTTP*, observou-se que, embora seja uma solução amplamente utilizada devido à sua compatibilidade com a infraestrutura *web*, apresenta limitações significativas em termos de eficiência e escalabilidade. O protocolo *HTTP*, que opera sobre *TCP*, garante a entrega confiável dos dados mas, à custa de um maior *overhead* e utilização de largura de banda.

A segunda etapa introduziu o conceito de *streaming* adaptativo com *DASH*. Esta abordagem demonstrou ser mais eficiente e flexível, permitindo a adaptação dinâmica da qualidade do vídeo às condições da rede e às capacidades do dispositivo do cliente. O *DASH* mostrou-se particularmente vantajoso em cenários com múltiplos clientes e condições de rede variáveis, oferecendo uma solução mais escalável em comparação com o *streaming* *HTTP* direto.

Na terceira etapa, foram exploradas soluções baseadas em *UDP*, tanto para *unicast* (*RTP/RTCP*) quanto para *Multicast* (*SAP*). O *streaming Unicast* via *RTP/RTCP* ofereceu uma alternativa mais leve ao *HTTP*, embora ainda enfrentasse desafios de escalabilidade para um grande número de clientes. Por outro lado, o *streaming Multicast* via *SAP* demonstrou ser a solução mais eficiente em termos de utilização de largura de banda e escalabilidade para cenários com muitos clientes simultâneos.

Comparando as diferentes abordagens, percebemos que o *streaming* *HTTP* é o mais simples de implementar e amplamente suportado, mas menos eficiente em termos de utilização de recursos. Vimos que o *DASH* oferece um bom equilíbrio entre adaptabilidade, eficiência e compatibilidade com infraestruturas *web* existentes. O *streaming Unicast* via *RTP/RTCP* é mais eficiente que o *HTTP*, mas ainda enfrenta desafios de escalabilidade. O *streaming Multicast* via *SAP* é a solução mais eficiente e escalável, mas requer suporte de rede específico e é mais complexo de implementar e gerir.

Este projeto destacou a importância de escolher a solução de *streaming* adequada com base nos requisitos específicos de cada aplicação, considerando fatores como o número de clientes esperados, as condições de rede, a necessidade de adaptabilidade e os recursos disponíveis. Além disso, o trabalho proporcionou uma experiência prática com ferramentas e tecnologias relevantes no campo de *streaming de média*, incluindo *FFmpeg*, *VLC*, *Wireshark* e o emulador CORE.

Em suma, este projeto ofereceu uma visão abrangente das tecnologias de *streaming* atuais, expondo as

respetivas vantagens e desafios, como também preparou nos para enfrentar os desafios de design e implementação de sistemas de *streaming* escaláveis.