



Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Engenharia de Serviços em Rede

Ano lectivo 2023/2024

TP1 - *Streaming* de áudio e vídeo a pedido e em tempo real

Grupo PL11

Miguel Silva Pinto, PG54105

Orlando José da Cunha Palmeira, PG54123

Pedro Miguel Castilho Martins, PG54146

12 de outubro de 2023

Índice

1	Introdução	1
2	Etapa 1 - <i>Streaming</i> HTTP simples sem adaptação dinâmica de débito	2
2.1	Questão 1	2
3	Etapa 2 - <i>Streaming</i> adaptativo sobre HTTP (MPEG-DASH)	5
3.1	Questão 2	5
3.2	Questão 3	6
3.3	Questão 4	8
4	Etapa 3 - <i>Streaming</i> RTP/RTCP <i>unicast</i> sobre UDP e <i>multicast</i> com anúncios SAP	9
4.1	Questão 5	9
5	Conclusão	11

Índice de figuras

2.1	Topologia utilizada	2
2.2	Detalhes do videoA.mp4	3
2.3	Estatísticas do tráfego gerado pelo servidor com 3 clientes	3
2.4	Tráfego no servidor com 1 cliente	3
2.5	Tráfego no servidor com 2 clientes	3
2.6	Tráfego no servidor com 3 clientes	4
3.1	Pilha protocolar	5
3.2	Parte do ficheiro <i>video_manifest.mpd</i>	5
3.3	Topologia com débito limitado para a Bela	6
3.4	Resolução da <i>stream</i> (Alladin vs Bela)	7
3.5	Pedidos HTTP entre os clientes e o VStreamer	7
4.1	Conversações Unicast	9
4.2	Conversações Multicast	9

1 Introdução

Este relatório foi elaborado no âmbito da UC de Engenharia de Serviços em Rede e aborda os resultados e análises das diferentes aplicações de diversos cenários de *streaming* de áudio e vídeo a pedido e em tempo real.

O trabalho teve como objetivo compreender as opções disponíveis em termos de pilha protocolar e as suas diferenças conceptuais, além da familiarização com formatos multimédia e técnicas de empacotamento através do uso de ferramentas *open-source* como *Wireshark*, *FFmpeg* e *VideoLAN VLC*.

O relatório está estruturado em três etapas principais. Na primeira etapa, focamo-nos no *streaming* de áudio e vídeo simples usando o protocolo HTTP, sem adaptação dinâmica de débito. Nesta etapa, utilizámos o VLC como servidor de *streaming* e como cliente. Na segunda etapa, introduzimos a adaptação dinâmica de débito e exploramos o uso de técnicas de *streaming* adaptativo, especificamente o *Dynamic Adaptive Streaming over HTTP* (DASH). Por fim, na terceira etapa, exploramos protocolos de *streaming* sobre UDP (*unicast* e *multicast*).

Ao longo do trabalho prático, realizamos diversas experiências em que analisamos o desempenho e escalabilidade de diferentes cenários de *streaming*. O relatório fornece um relato das tarefas realizadas e uma discussão dos resultados obtidos.

2 Etapa 1 - *Streaming* HTTP simples sem adaptação dinâmica de débito

2.1 Questão 1

Capture três pequenas amostras de tráfego no *link* de saída do servidor, respectivamente com 1 cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffmpeg). 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).

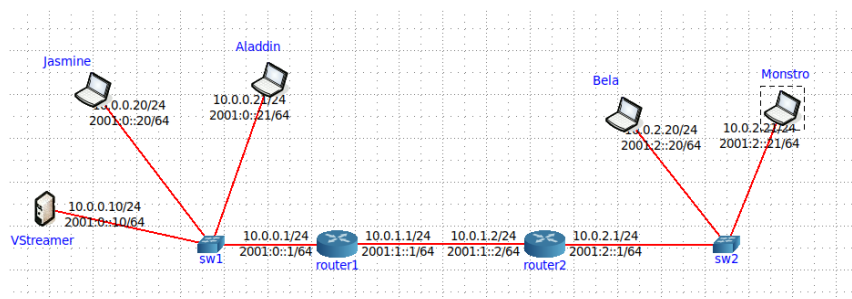


Figura 2.1: Topologia utilizada

Neste exercício, foi testado o *streaming* HTTP sem adaptação dinâmica de débito, feito pelo servidor VStreamer atendendo vários clientes.

Através da análise do comando `ffmpeg -i videoA.mp4`, determinamos que o bitrate do vídeo é de 18 kbps (18000 bps) como está demonstrado na figura 2.2, logo, teoricamente, a taxa em bps necessária para transmitir o vídeo para 3 clientes seria de $18 \times 3 = 54 \text{ kbps} = 54000 \text{ bps}$.

```

core@xubuncore:~/Desktop/Trabalho-ESR-2023-2024/TP1/videos$ ffmpeg -i videoA.mp4
ffmpeg version 4.2.7-0ubuntu0.1 Copyright (C) 2000-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-1ubuntu1-20.04.1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incd
  ire=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl --disable-stripping --enable-auresample --disable-filterresna
  mple --enable-avizynth --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs
  2b --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-libop
  enjpe --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librsync --enable-librubberband --enable-libshine
  --enable-lisnappy --enable-libsoxr --enable-lspspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-li
  bvidstab --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --ena
  ble-libxvid --enable-libzmq --enable-libzbi --enable-lv2 --enable-omx --enable-opengl --enable-opencore --enable-opengl -
  --enable-sdl2 --enable-libcd1394 --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
  encoder          : Lavf58.76.100
Duration: 00:00:05.35, start: 0.000000, bitrate: 21 kb/s
Stream #0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 200x150, 18 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 t
  be (default)
Metadata:
  handler_name     : VideoHandler
At least one output file must be specified
core@xubuncore:~/Desktop/Trabalho-ESR-2023-2024/TP1/videos$

```

Figura 2.2: Detalhes do videoA.mp4

Seguidamente, analisamos o tráfego gerado no servidor com três clientes, no Wireshark, tendo verificado que na realidade a taxa de transferência efetiva foi de 117 kbps.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	576	100.0	392241	456 k	0	0	0
Ethernet	100.0	576	2.1	8064	9381	0	0	0
Internet Protocol Version 4	100.0	576	2.9	11520	13 k	0	0	0
Transmission Control Protocol	100.0	576	95.0	372657	433 k	507	270813	315 k
Hypertext Transfer Protocol	12.0	69	25.8	101298	117 k	69	101298	117 k

Figura 2.3: Estatísticas do tráfego gerado pelo servidor com 3 clientes

Analisando no Wireshark o tráfego gerado pelo servidor com um crescente número de clientes, obtivemos os seguintes resultados.

Com um cliente, o portátil "Jasmine" através do VLC, é gerado apenas um fluxo de dados.

Ethernet - 3		IPv4 - 2		TCP - 1		UDP	
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B
10.0.0.20	46970	10.0.0.10	8080	104	72 k	52	3432

Figura 2.4: Tráfego no servidor com 1 cliente

Acrescentando mais um cliente, o portátil "Bela" por Firefox, são gerados dois fluxos de dados.

Ethernet - 4		IPv4 - 3		TCP - 2		UDP	
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B
10.0.0.20	46970	10.0.0.10	8080	354	242 k	177	11 k
10.0.0.20	55292	10.0.0.10	8080	177	11 k	177	230 k

Figura 2.5: Tráfego no servidor com 2 clientes

E finalmente, com três clientes ativos, são gerados três fluxos de dados.

Ethernet - 4		IPv4 - 4		IPv6 - 1		TCP - 3		UDP							
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A		
10.0.0.20	46970	10.0.0.10	8080	192	130 k	96	6336	96	124 k	0.000000	6.8764		7371		
10.0.2.20	48686	10.0.0.10	8080	192	130 k	96	6336	96	124 k	0.000060	6.8764		7371		
10.0.2.21	54544	10.0.0.10	8080	192	130 k	96	6336	96	124 k	0.000105	6.8764		7371		

Figura 2.6: Tráfego no servidor com 3 clientes

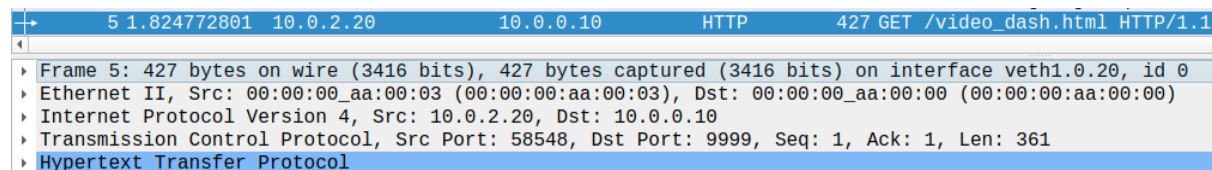
Com estes resultados podemos afirmar que é criada uma nova conexão por cada cliente com esta arquitetura, o que significa que esta solução apresenta uma escalabilidade linear em que tem de ser gerado um novo fluxo de dados por cada cliente que pretende receber o conteúdo do servidor. Esta característica afeta negativamente o sistema uma vez que todo o trabalho necessário para receber o conteúdo multimédia é posto do lado do servidor, requerendo muita largura de banda do servidor para os pedidos de vários clientes.

3 Etapa 2 - *Streaming* adaptativo sobre HTTP (MPEG-DASH)

3.1 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.

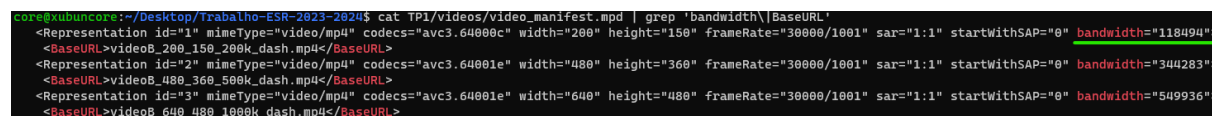
A pilha protocolar utilizada é HTTP/TCP/IP/Ethernet, conforme demonstrado na figura seguinte onde se apresentam todos os protocolos utilizados num dos pacotes HTTP capturados:



5	1.824772801	10.0.2.20	10.0.0.10	HTTP	427 GET /video_dash.html HTTP/1.1
Frame 5: 427 bytes on wire (3416 bits), 427 bytes captured (3416 bits) on interface veth1.0.20, id 0					
Ethernet II, Src: 00:00:00_aa:00:03 (00:00:00:aa:00:03), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)					
Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.0.10					
Transmission Control Protocol, Src Port: 58548, Dst Port: 9999, Seq: 1, Ack: 1, Len: 361					
Hypertext Transfer Protocol					

Figura 3.1: Pilha protocolar

A largura de banda mínima teórica necessária para poder apresentar o vídeo de **menor resolução** (200×150) é de **118494 bps**, como é indicado pelo ficheiro *video_manifest.mpd* e sublinhado na figura 3.2.



```
core@xubuncore:~/Desktop/trabalho-ESR-2023-2024$ cat TP1/videos/video_manifest.mpd | grep 'bandwidth|BaseURL'
<Representation id="1" mimeType="video/mp4" codecs="avc3.64000c" width="200" height="150" frameRate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="118494">
<BaseURL>videoB_200_150_200k_dash.mp4</BaseURL>
<Representation id="2" mimeType="video/mp4" codecs="avc3.64001e" width="480" height="360" frameRate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="344283">
<BaseURL>videoB_480_360_500k_dash.mp4</BaseURL>
<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="480" frameRate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="549936">
<BaseURL>videoB_640_480_1000k_dash.mp4</BaseURL>
```

Figura 3.2: Parte do ficheiro *video_manifest.mpd*

Neste ficheiro são também apresentadas as larguras de banda mínimas teóricas para serem enviados os vídeos com **melhor resolução**, sendo necessária uma largura de banda mínima de **344283 bps** para apresentar o vídeo em 480×360 e **549936 bps** para 640×480.

3.2 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.

Segundo o ficheiro MPD, para que o portátil **Alladin** exiba o vídeo com maior resolução, é necessário ter uma largura de banda superior ao limite de **549936 bps** como foi demonstrado na pergunta anterior. No entanto nas nossas experiências o portátil Alladin acabava por reduzir a sua resolução para 480×360 , mesmo não tendo um limite de *bandwidth* na ligação com o servidor, como se pode verificar na figura 3.5. Isto poderá ter acontecido devido ao facto de o servidor estar a servir 2 clientes em simultâneo.

Já o portátil **Bela** para apresentar o vídeo de menor resolução precisa de ter uma largura de banda entre o intervalo de **[118494, 344283] bps**, segundo o ficheiro MPD. No entanto pelos nossos testes, o vídeo continuou a não ser reproduzido e foi necessário aumentar a *bandwidth* da ligação até 600 kbps. Com este débito o portátil Bela conseguiu reproduzir o vídeo na menor resolução (200×150).

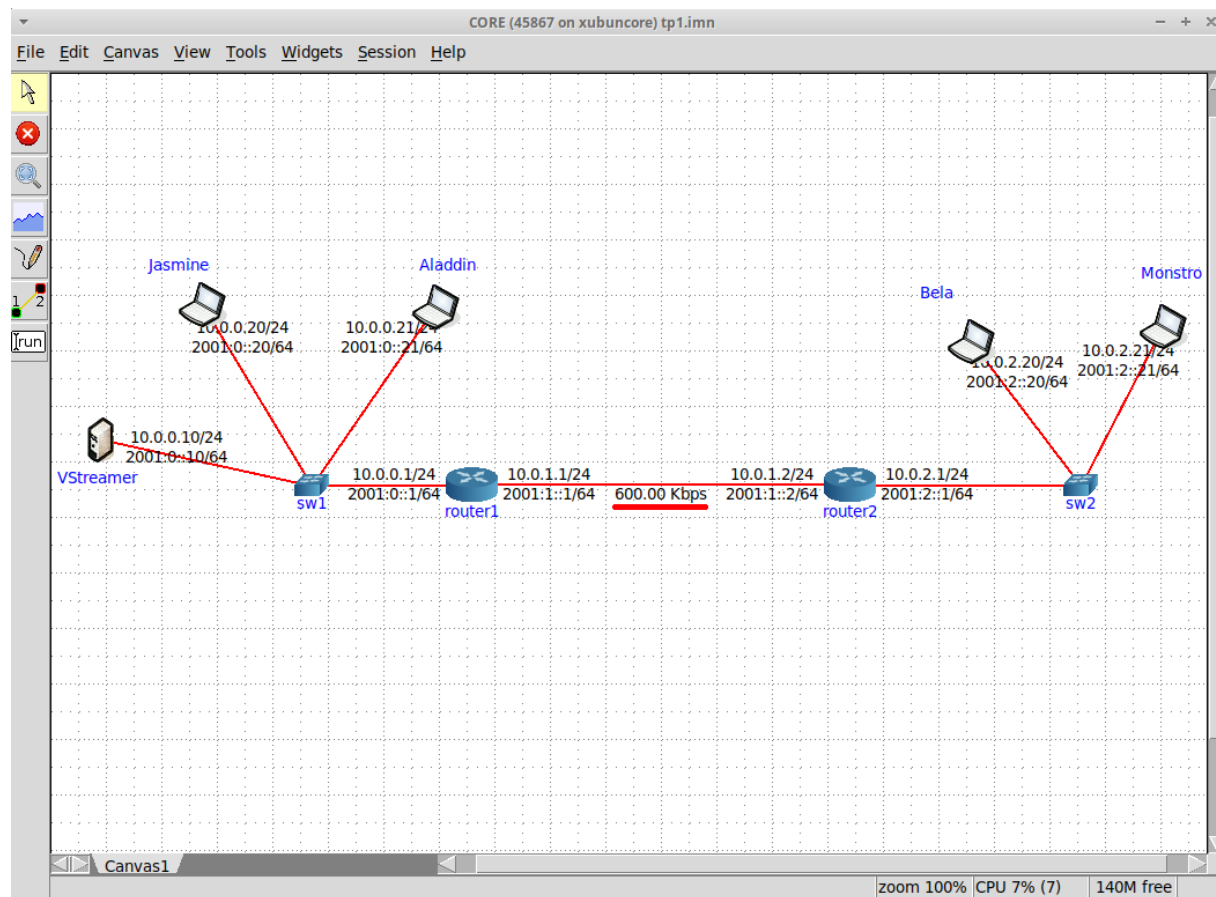


Figura 3.3: Topologia com débito limitado para a Bela

Nas seguintes capturas de ecrã podemos ver como o *streaming* adaptativo se comporta durante a transmissão do vídeo para 2 clientes.

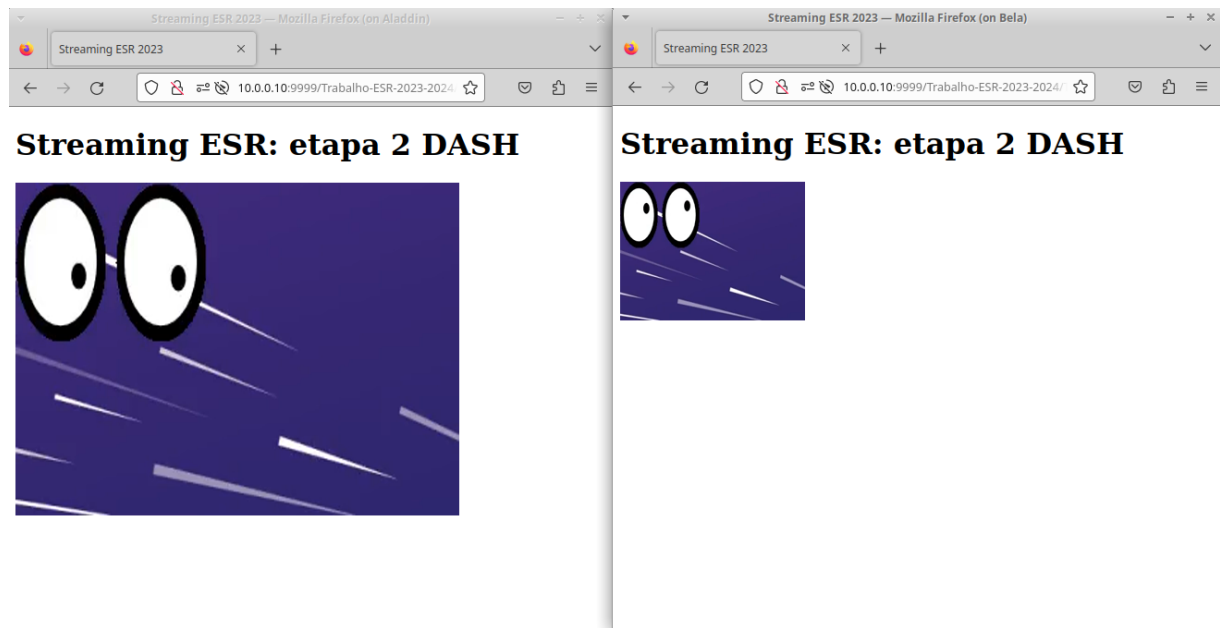


Figura 3.4: Resolução da *stream* (Alladin vs Bela)

No.	Time	Source	Destination	Protocol	Length	Info
20617	7.298437419	10.0.0.10	10.0.0.21	MP4	1391	
20635	7.412275633	10.0.0.21	10.0.0.10	HTTP	476	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_640_480_1000k_dash.mp4 HTTP/1.1
21500	7.414757015	10.0.0.10	10.0.0.21	MP4	1391	
21513	7.516314290	10.0.0.21	10.0.0.10	HTTP	476	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_640_480_1000k_dash.mp4 HTTP/1.1
22412	7.519402740	10.0.0.10	10.0.0.21	MP4	1391	
22427	7.599512346	10.0.0.21	10.0.0.10	HTTP	476	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_640_480_1000k_dash.mp4 HTTP/1.1
23304	7.617662391	10.0.0.10	10.0.0.21	MP4	1391	
23390	8.364042179	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
23556	9.958571116	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_200_150_200k_dash.mp4 HTTP/1.1
23603	10.267044767	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
24141	10.269727877	10.0.0.10	10.0.0.21	MP4	1450	
24148	10.282607138	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
24686	10.284732138	10.0.0.10	10.0.0.21	MP4	1450	
24697	10.306245389	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
25224	10.332513183	10.0.0.10	10.0.0.21	MP4	1450	
25231	10.347181572	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
25764	10.361003404	10.0.0.10	10.0.0.21	MP4	1450	
25771	10.378046640	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
26312	10.395999310	10.0.0.10	10.0.0.21	MP4	1450	
26319	10.410402182	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
26853	10.420028599	10.0.0.10	10.0.0.21	MP4	1450	
26860	10.441754509	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
27400	10.458324794	10.0.0.10	10.0.0.21	MP4	1450	
27410	10.489684586	10.0.0.21	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
27950	10.500867050	10.0.0.10	10.0.0.21	MP4	1450	
29365	22.877299917	10.0.0.10	10.0.2.20	MP4	1257	
29663	25.785267941	10.0.0.10	10.0.2.20	MP4	1391	
30114	33.565427341	10.0.2.20	10.0.0.10	HTTP	472	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_640_480_1000k_dash.mp4 HTTP/1.1
30152	33.916949474	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
30165	33.923919980	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_200_150_200k_dash.mp4 HTTP/1.1
30944	40.890527674	10.0.0.10	10.0.2.20	MP4	1257	
31869	52.344884994	10.0.2.20	10.0.0.10	HTTP	472	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_640_480_1000k_dash.mp4 HTTP/1.1
31882	52.352189087	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_200_150_200k_dash.mp4 HTTP/1.1
31897	52.360855978	10.0.2.20	10.0.0.10	HTTP	473	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_480_360_500k_dash.mp4 HTTP/1.1
32769	60.301644579	10.0.0.10	10.0.2.20	MP4	1257	
33095	65.514572795	10.0.2.20	10.0.0.10	HTTP	475	GET /Trabalho-ESR-2023-2024/TP1/videos/videoB_200_150_200k_dash.mp4 HTTP/1.1
33372	68.404435063	10.0.0.10	10.0.2.20	MP4	1257	

Figura 3.5: Pedidos **HTTP** entre os clientes e o VStreamer

Na figura 3.5 podemos ver a **azul** os pedidos entre o servidor e o cliente Alladin. O cliente a certa altura pede ao servidor para baixar a resolução do vídeo para a resolução média disponível, apesar de não haver nenhuma limitação imposta.

Os pedidos rodeados a **vermelho** são pedidos entre o servidor e o cliente Bela onde podemos ver que o cliente Bela necessita de pedir a resolução mais baixa para conseguir reproduzir o vídeo.

3.3 Questão 4

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

O DASH, ou *Dynamic Adaptive Streaming over HTTP*, é um protocolo de *streaming* de vídeo que permite a entrega adaptativa de conteúdo multimédia pela Internet. O protocolo permite que um servidor de media forneça diferentes versões de um vídeo em vários níveis de qualidade e permite que o cliente escolha a melhor versão para reprodução com base nas condições da rede e nas capacidades do dispositivo, através da informação contida no ficheiro MPD.

O ficheiro MPD (Media Presentation Description) é um ficheiro XML que atua como um manifesto ou guia para o conteúdo de vídeo disponível. O MPD descreve as diferentes versões do vídeo, a resolução, o *bitrate* e os URLs dos segmentos de vídeo para cada versão. Ele também contém informações sobre as regras de adaptação, como a lógica para selecionar a qualidade apropriada com base nas condições de rede do cliente.

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

4.1 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.

No cenário *unicast*, os dados são enviados apenas para o endereço do portátil Monstro e cada cliente que pretenda receber o conteúdo multimédia receberá uma cópia dos dados única, o que exige muita largura de banda do lado do servidor e acarreta um aumento de carga na infraestrutura da rede à medida que o número de destinatários aumenta.

A seguinte imagem demonstra as conversações ocorridas em *Unicast* com 4 clientes.

Ethernet · 4				IPv4 · 4				IPv6	TCP	UDP · 8							
Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A			
10.0.0.10	33970	10.0.0.20	5555	321	281 kB	0	321	281 kB	0	0 bytes	0.000000	11.2547	199 kbps	0 bits/s			
10.0.0.10	42669	10.0.0.21	5555	320	280 kB	1	320	280 kB	0	0 bytes	0.023398	11.1978	200 kbps	0 bits/s			
10.0.0.10	36112	10.0.0.22	5555	320	280 kB	3	320	280 kB	0	0 bytes	0.039249	11.1970	200 kbps	0 bits/s			
10.0.0.10	48861	10.0.0.23	5555	320	280 kB	2	320	280 kB	0	0 bytes	0.028915	11.2044	200 kbps	0 bits/s			
10.0.0.10	33971	10.0.0.20	5556	2	140 bytes	7	2	140 bytes	0	0 bytes	4.150573	5.0489	221 bits/s	0 bits/s			
10.0.0.10	42670	10.0.0.21	5556	2	140 bytes	5	2	140 bytes	0	0 bytes	4.128825	5.0019	223 bits/s	0 bits/s			
10.0.0.10	36113	10.0.0.22	5556	2	140 bytes	6	2	140 bytes	0	0 bytes	4.139103	5.0020	223 bits/s	0 bits/s			
10.0.0.10	48862	10.0.0.23	5556	2	140 bytes	4	2	140 bytes	0	0 bytes	3.982552	5.0482	221 bits/s	0 bits/s			

Figura 4.1: Conversações *Unicast*

Como podemos ver pela figura 4.1, por **cada cliente** é necessário gerar **um fluxo de dados distinto** que ocupa **200 kbps** da largura de banda do servidor, apresentando uma **fraca escalabilidade**.

No cenário *multicast*, o conteúdo multimédia é encaminhado na rede **apenas uma vez** e os dados são bifurcados para os destinatários, economizando largura de banda e tráfego na rede, tal como se pode constatar na figura 4.2.

Ethernet · 2				IPv4 · 2				IPv6	TCP	UDP · 3							
Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A			
10.0.0.10	37641	224.0.0.200	5555	321	281 kB	0	321	281 kB	0	0 bytes	0.000000	11.2442	200 kbps	0 bits/s			
10.0.0.10	37642	224.0.0.200	5556	3	210 bytes	2	3	210 bytes	0	0 bytes	0.998213	10.0440	167 bits/s	0 bits/s			
10.0.0.10	55144	224.2.127.254	9875	3	1 kB	1	3	1 kB	0	0 bytes	0.997844	10.0012	878 bits/s	0 bits/s			

Figura 4.2: Conversações *Multicast*

Nesta amostra, o servidor utiliza ***Multicast*** para também servir **4 clientes**, mas apenas **um fluxo de dados** foi gerado a partir do servidor, ao contrário do ***Unicast***, ocupando apenas 200 kbps da largura de banda do servidor independentemente do número de clientes.

Esta vantagem é possível devido à utilização de um endereço ***multicast*** que permite a transmissão de dados de um único remetente para vários recetores simultaneamente, sem a necessidade de replicar os dados para cada destinatário separadamente.

Em suma, o ***multicast*** supera o ***unicast*** tanto em **escalabilidade** como no **tráfego de rede** quando se trata de distribuir conteúdo para vários utilizadores, uma vez que o ***unicast*** rapidamente se torna ineficiente, consumindo desnecessariamente largura de banda e recursos da rede com um número alargado de clientes.

5 Conclusão

A realização deste trabalho implicou a análise da implementação de diversas estratégias de *streaming* de áudio e vídeo a pedido e em tempo real.

Na primeira etapa, verificamos que a utilização de *streaming* HTTP simples, sem adaptação de taxa de bits, revelou-se inviável devido à sobrecarga imposta ao serviço pelo aumento do número de clientes.

Na segunda etapa, tivemos a oportunidade de observar o funcionamento do DASH, acompanhando a alteração da resolução dos vídeos para se ajustarem à largura de banda disponível no momento. Nesta etapa, deparamo-nos com alguns problemas, uma vez que os vídeos não conseguiam ser reproduzidos com as taxas de bits mínimas teoricamente especificadas no ficheiro MPD, o que nos levou a ajustar esses valores para níveis manifestamente mais elevados.

Na última fase, procedemos à comparação entre o *streaming* RTP/RTCP unicast sobre UDP e o multicast com anúncios SAP. Verificamos que a opção de *streaming* em *multicast* supera a *unicast* em vários aspetos, nomeadamente em termos de escalabilidade e utilização de recursos da rede.