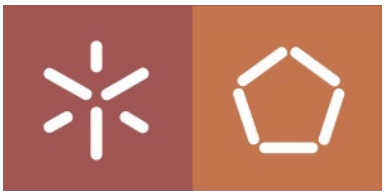

Engenharia de Serviços em Rede

22/23

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

Grupo 24

13 de outubro de 2022

João Neves (PG50497) Patrícia Pereira (PG50673) Meriem Khammassi (A85829)

Índice

Lista de Figuras

1 Etapa 1. Streaming HTTP simples sem adaptação dinâmica de débito

1.1 Questão 1

A) Capture três pequenas amostras de tráfego no link de saída, respetivamente com um cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay).

As seguintes figuras são as amostras de tráfego capturado no link de saída do servidor.

Denota-se que, no link de saída do servidor com vários clientes, em cada amostra, conseguimos identificar os diferentes clientes e o Servidor, *VStreamer*.

No.	Time	Source	Destination	Protocol	Length	Info
29	45.203035526	10.0.0.10	10.0.0.20	TCP	74	8080 → 50838 [SYN, ACK] Seq=0 Ack=1 Win=65160 Le
30	45.203047579	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 T
31	45.203103192	10.0.0.20	10.0.0.10	HTTP	200	GET / HTTP/1.1
32	45.203108723	10.0.0.10	10.0.0.20	TCP	66	8080 → 50838 [ACK] Seq=1 Ack=135 Win=65152 Len=(
33	45.226852716	10.0.0.10	10.0.0.20	TCP	169	8080 → 50838 [PSH, ACK] Seq=1 Ack=135 Win=65152
34	45.226884550	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=104 Win=64256 Ler
35	45.226904157	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=104 Ack=135 Win=65152 Ler
36	45.226904319	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=1552 Ack=135 Win=65152 Le
37	45.226904416	10.0.0.10	10.0.0.20	TCP	543	8080 → 50838 [PSH, ACK] Seq=3000 Ack=135 Win=65
38	45.226914860	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=1552 Win=64128 Le
39	45.226915595	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=3000 Win=63488 Le
40	45.226916186	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=3477 Win=63232 Le
41	45.226929677	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=3477 Ack=135 Win=65152 Le
42	45.226929803	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=4925 Ack=135 Win=65152 Le
43	45.226929900	10.0.0.10	10.0.0.20	TCP	1510	8080 → 50838 [PSH, ACK] Seq=6373 Ack=135 Win=65
44	45.226957114	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=4925 Win=62464 Le
45	45.226958348	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=135 Ack=6373 Win=61824 Le

Figura 1: Amostra de tráfego com 1 cliente

No.	Time	Source	Destination	Protocol	Length	Info
40	0.467438146	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=25169 Win=643 Len=(
41	0.467439339	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=26617 Win=637 Len=(
42	0.467439936	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=28061 Win=631 Len=(
43	0.718323912	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=28061 Ack=1 Win=509 Len=1
44	0.718324393	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=29509 Ack=1 Win=509 Len=1
45	0.718324519	10.0.0.10	10.0.0.20	TCP	1493	8080 → 50838 [PSH, ACK] Seq=30957 Ack=1 Win=509
46	0.718358637	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=29509 Win=665 Len=(
47	0.718360127	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=30957 Win=684 Len=(
48	0.718360796	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=32384 Win=678 Len=(
49	0.996359449	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=32384 Ack=1 Win=509 Len=1
50	0.996359872	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=33832 Ack=1 Win=509 Len=1
51	0.996359994	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=35280 Ack=1 Win=509 Len=1
52	0.996360070	10.0.0.10	10.0.0.20	TCP	426	8080 → 50838 [PSH, ACK] Seq=36728 Ack=1 Win=509
53	0.996392928	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=33832 Win=689 Len=(
54	0.996394227	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=35280 Win=684 Len=(
55	0.996394814	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=36728 Win=678 Len=(

Figura 2: Amostra de tráfego com 2 clientes

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=1 Ack=1 Win=509
2	0.000000490	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=1449 Ack=1 Win=
3	0.000000606	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=2897 Ack=1 Win=
4	0.000000685	10.0.0.10	10.0.0.20	TCP	405	8080 → 50838 [PSH, ACK] Seq=4345 Ack=1
5	0.000033990	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=1449 Win=
6	0.000035231	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=2897 Win=
7	0.000035845	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=4345 Win=
8	0.000036432	10.0.0.20	10.0.0.10	TCP	66	50838 → 8080 [ACK] Seq=1 Ack=4684 Win=
9	0.000082598	10.0.0.10	10.0.2.21	TCP	1514	8080 → 60938 [ACK] Seq=1 Ack=1 Win=507
10	0.000082946	10.0.0.10	10.0.2.21	TCP	1514	8080 → 60938 [ACK] Seq=1449 Ack=1 Win=
11	0.000083046	10.0.0.10	10.0.2.21	TCP	1514	8080 → 60938 [ACK] Seq=2897 Ack=1 Win=
12	0.000083131	10.0.0.10	10.0.2.21	TCP	405	8080 → 60938 [PSH, ACK] Seq=4345 Ack=1
13	0.000137774	10.0.2.21	10.0.0.10	TCP	66	60938 → 8080 [ACK] Seq=1 Ack=1449 Win=
14	0.000138783	10.0.2.21	10.0.0.10	TCP	66	60938 → 8080 [ACK] Seq=1 Ack=2897 Win=
15	0.000139438	10.0.2.21	10.0.0.10	TCP	66	60938 → 8080 [ACK] Seq=1 Ack=4345 Win=
16	0.000140011	10.0.2.21	10.0.0.10	TCP	66	60938 → 8080 [ACK] Seq=1 Ack=4684 Win=
17	0.263360003	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=4684 Ack=1 Win=
18	0.263361097	10.0.0.10	10.0.0.20	TCP	1514	8080 → 50838 [ACK] Seq=6132 Ack=1 Win=
19	0.263361384	10.0.0.10	10.0.0.20	TCP	1510	8080 → 50838 [PSH, ACK] Seq=7580 Ack=1

Figura 3: Amostra de tráfego com 3 clientes

B) Identifique a taxa em bps necessária (usando o ffmpeg -i vídeoA.mp4 e/ou o próprio wireshark), o encapsulamento usado e o número total de fluxos gerados.

A partir da primeira amostra de tráfego no link de saída com um cliente, conseguimos identificar, utilizando o comando *ffmpeg -i vídeoA.mp4* que a taxa teórica necessária é de 25000bps, enquanto, a taxa real, obtida no wireshark é de 63000bps.

```
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoA.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.29.100
  Duration: 00:00:03.50, start: 0.000000, bitrate: 29 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 160x100,
25 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
    Metadata:
      handler name    : VideoHandler
```

Figura 4: Taxa teórica obtida com o comando ffmpeg

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	460	100.0	300184	141 k	0	0	0
Ethernet	100.0	460	2.1	6440	3036	0	0	0
Internet Protocol Version 6	0.9	4	0.1	160	75	0	0	0
User Datagram Protocol	0.4	2	0.0	16	7	0	0	0
Multicast Domain Name System	0.4	2	0.0	90	42	2	90	42
Open Shortest Path First	0.2	1	0.0	36	16	1	36	16
Internet Control Message Protocol v6	0.2	1	0.0	16	7	1	16	7
Internet Protocol Version 4	98.7	454	3.0	9080	4281	0	0	0
Transmission Control Protocol	96.7	445	94.6	283894	133 k	444	283728	133 k
Hypertext Transfer Protocol	0.2	1	0.0	134	63	1	134	63
Open Shortest Path First	2.0	9	0.1	396	186	9	396	186
Address Resolution Protocol	0.4	2	0.0	56	26	2	56	26

Figura 5: Taca teórica real com o Wireshark

A diferença entre as duas taxas deve-se à existência de picos no débito na captura. Deste modo a taxa capturada é maior no início e converge para o valor teórico à medida que o tempo passa.

Nas capturas de tráfego obtidas observamos o seguinte:

- Na camada física da pilha protocolar é utilizado o Protocolo Ethernet.
- Na camada de Rede é utilizado o IPV4 (*Internet Protocol Version 4*).
- Na camda de Transporte é utilizado TCP (*Transmission Control Protocol).*
- E, na camada de Aplicação, o Http (*Hypertext Trasnfer Protocol*).

O último protocolo mencionado, é apenas identificado no cabeçalho do primeiro pacote enviado, enquanto que os restantes pacotes transportam o *payload*, resultante da fragmentação. Podemos concluir isto pela análise das duas figuras seguintes, onde o protocolo Http é identificado no primeiro pacote enviado, mas não nos restantes.

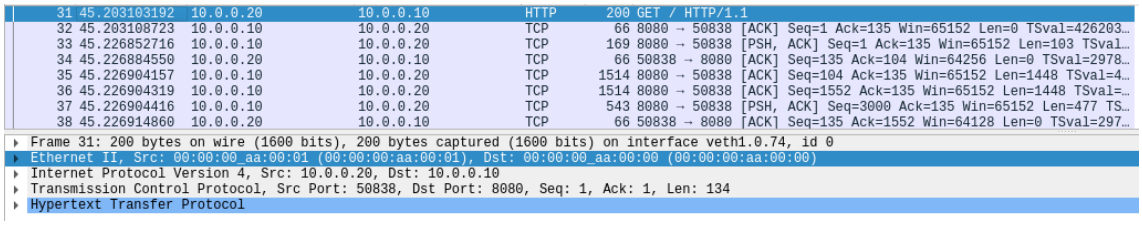

Figura 6: Primeiro pacote de uma transmissão HTTP

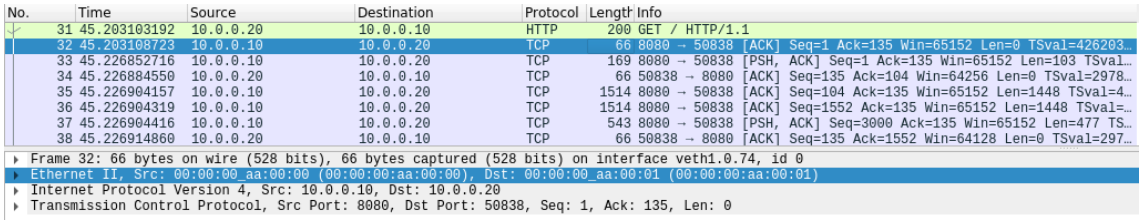

Figura 7: Segundo pacote de uma transmissão HTTP

Aplicando vários filtros de stream às capturas de tráfego no link de saída do servidor, com três clientes, retiramos que apenas existem três fluxos correspondendo a cada um dos clientes.

Nas figuras seguintes, podemos observar que, enquanto existe tráfego entre o cliente e o *VStreamer* aplicando os três primeiros filtros , isso não se verifica ao aplicar o 4°.

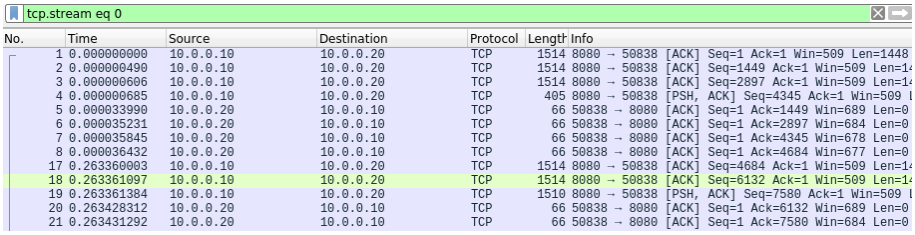

Figura 8: Tráfego de rede com filtro de fluxo 0

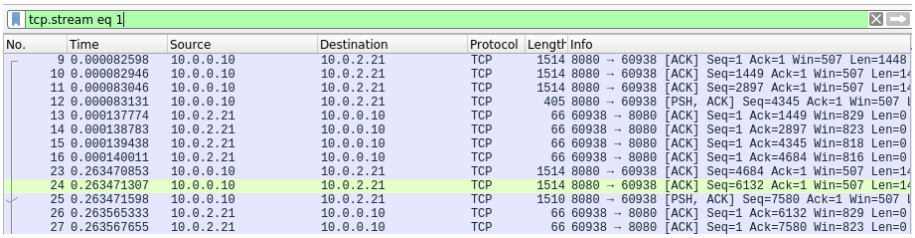

Figura 9: Tráfego de rede com filtro de fluxo 1

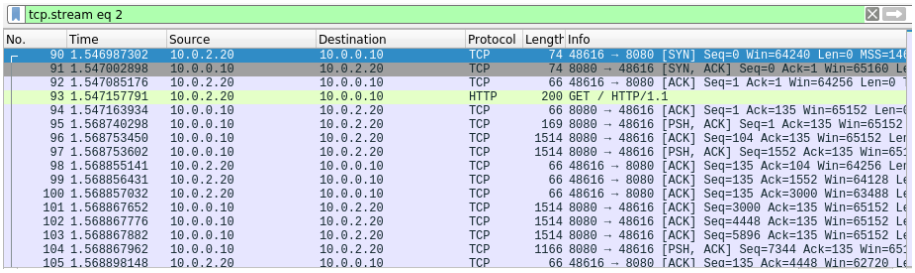

Figura 10: Tráfego de rede com filtro de fluxo 2

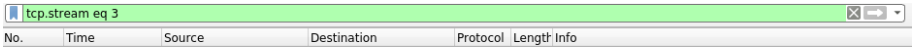

Figura 11: Tráfego de rede com filtro de fluxo 3

C) Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício.

Sendo que a transmissão é unicast, ou seja, ponto-a-ponto, há um aumento linear com o aumento do número de clientes a que o servidor transmite, pelo que, esta solução, não é escalável.

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

2.1 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.

Para obter a largura de banda, em bits por segundo, para que o cliente de streaming consiga receber o vídeo no Firefox, verificamos em primeiro lugar os valores do ficheiro *vídeo_manifest.mpd*.

```
</SegmentList>
<Representation id="1" mimeType="video/mp4" codecs="avc3.64000c" width="160" height="100" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="87237">
 <BaseURL>videoB_160_100_200k_dash.mp4</BaseURL>
 <SegmentList timescale="15360" duration="7680">
   <SegmentURL mediaRange="927-5238" indexRange="927-970"/>
   <SegmentURL mediaRange="5239-8971" indexRange="5239-5282"/>
   <SegmentURL mediaRange="8972-13336" indexRange="8972-9015"/>
   <SegmentURL mediaRange="13337-23555" indexRange="13337-13380"/>
   <SegmentURL mediaRange="23556-29752" indexRange="23556-23599"/>
   <SegmentURL mediaRange="29753-35864" indexRange="29753-29796"/>
  <SegmentURL mediaRange="35865-36348" indexRange="35865-35908"/>
 </SegmentList>
```

Figura 12: Vídeo com Resolução 160x100

```
<Representation id="2" mimeType="video/mp4" codecs="avc3.640014" width="320" height="200" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="208687">
 <BaseURL>videoB_320_200_500k_dash.mp4</BaseURL>
 <SegmentList timescale="15360" duration="7680">
   <SegmentURL mediaRange="928-10445" indexRange="928-971"/>
   <SegmentURL mediaRange="10446-19625" indexRange="10446-10489"/>
   <SegmentURL mediaRange="19626-30634" indexRange="19626-19669"/>
   <SegmentURL mediaRange="30635-56693" indexRange="30635-30678"/>
   <SegmentURL mediaRange="56694-71417" indexRange="56694-56737"/>
   <SegmentURL mediaRange="71418-86063" indexRange="71418-71461"/>
  <SegmentURL mediaRange="86064-86952" indexRange="86064-86107"/>
 </SegmentList>
```

Figura 13: Vídeo com Resolução 320x200

```
<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e" width="640" height="400" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="483852">
 <BaseURL>videoB_640_400_1000k_dash.mp4</BaseURL>
 <SegmentList timescale="15360" duration="7680">
   <SegmentURL mediaRange="927-21634" indexRange="927-970"/>
   <SegmentURL mediaRange="21635-43658" indexRange="21635-21678"/>
   <SegmentURL mediaRange="43659-73195" indexRange="43659-43702"/>
   <SegmentURL mediaRange="73196-135374" indexRange="73196-73239"/>
   <SegmentURL mediaRange="135375-167341" indexRange="135375-135418"/>
   <SegmentURL mediaRange="167342-200051" indexRange="167342-167385"/>
  <SegmentURL mediaRange="200052-201604" indexRange="200052-200095"/>
 </SegmentList>
```

Figura 14: Vídeo com Resolução 640x400

Após analisar estes valores, verificamos que:

- O vídeo com resolução 160x100 requer de 87237bps.
- O vídeo com resolução 320x200 requer de 208687bps.
- E o vídeo com resolução 640x400 requer de 483852bps.

Na captura de tráfego obtida no Wireshark, conseguimos reter que:

- Na camada física da pilha protocolar é utilizado o Protocolo Ethernet.
- Na camada de Rede é utilizado o IPV4.
- Na camada de Transporte é utilizado TCP.
- E, na camada de Aplicação, o Http.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▾ Frame	100.0	27	100.0	12980	4385	0	0	0
▾ Ethernet	100.0	27	2.9	378	127	0	0	0
▾ Internet Protocol Version 4	100.0	27	4.2	540	182	0	0	0
▾ Transmission Control Protocol	100.0	27	92.9	12062	4075	0	0	0
▸ Hypertext Transfer Protocol	100.0	27	12056.3	1564902	528 k	14	4670	1577

Figura 15: Captura de tráfego Wireshark

2.2 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.

Na topologia construída no início do trabalho prático, quando era pedido o envio de um vídeo, era enviado com maior resolução.

Para que fosse possível o cliente no Portátil Bela exibir o vídeo de menor resolução tivemos de alterar a topologia.

Essa mudança consistiu em adicionar um limite de largura de banda na ligação entre o Switch sw2 e o Portátil Bela de 100kbs. Assim garantimos que o portátil Bela só consegue receber o vídeo de menor resolução.

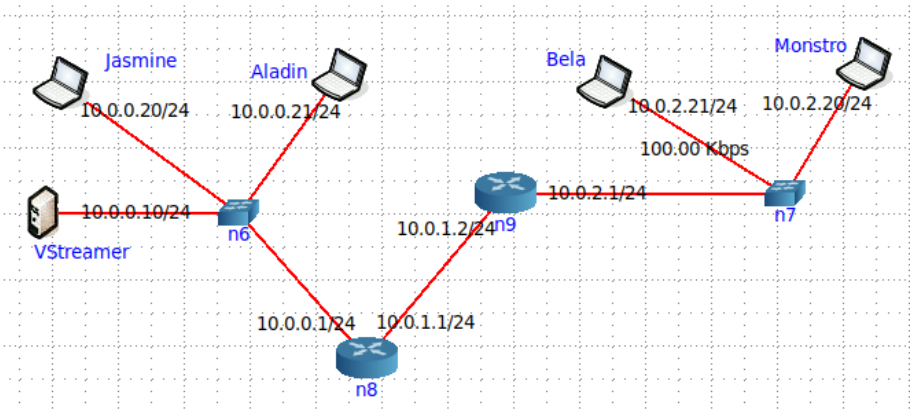

Figura 16: Topologia com limite de Largura de Banda

Para provar estes resultados, fizemos uma captura do tráfego.

Na imagem seguinte podemos concluir que:

- O Portátil Alladin de IP 10.0.0.21 fez um pedido do vídeo com resolução 640x400, vídeo de maior resolução e manteve-se a pedir esse.

- O portátil Bela com IP 10.0.2.21 fez um pedido do vídeo com resolução 640x400, vídeo de maior resolução e depois passou a pedir o de menor resolução 160x100 continuando a pedir esse.

No.	Time	Source	Destination	Protocol	Length	Info
19	20.385845246	10.0.0.10	10.0.2.21	HTTP	741	HTTP/1.1 404 Not Found (text/html)
26	21.468710979	10.0.2.21	10.0.0.10	HTTP	401	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
90	24.278460652	10.0.2.21	10.0.0.10	HTTP	400	GET /videoB_160_100_200k_dash.mp4 HTTP/1.1
128	25.298022629	10.0.0.10	10.0.2.21	MP4	419	
147	27.443576176	10.0.2.21	10.0.0.10	HTTP	401	GET /videoB_160_100_200k_dash.mp4 HTTP/1.1
183	28.418160468	10.0.0.10	10.0.2.21	MP4	419	
204	30.586717220	10.0.2.21	10.0.0.10	HTTP	402	GET /videoB_160_100_200k_dash.mp4 HTTP/1.1
240	31.567738832	10.0.0.10	10.0.2.21	MP4	419	
274	33.794043750	10.0.2.21	10.0.0.10	HTTP	402	GET /videoB_160_100_200k_dash.mp4 HTTP/1.1
312	34.785931240	10.0.0.10	10.0.2.21	MP4	419	
340	42.942475439	10.0.0.21	10.0.0.10	HTTP	381	GET /favicon.ico HTTP/1.1
342	42.942616237	10.0.0.10	10.0.0.21	HTTP	741	HTTP/1.1 404 Not Found (text/html)
349	43.738999401	10.0.0.21	10.0.0.10	HTTP	401	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
524	43.740023818	10.0.0.10	10.0.0.21	MP4	604	
536	43.840951279	10.0.0.21	10.0.0.10	HTTP	403	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
690	43.842300255	10.0.0.10	10.0.0.21	MP4	604	
707	43.904344735	10.0.0.21	10.0.0.10	HTTP	403	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
861	43.905566127	10.0.0.10	10.0.0.21	MP4	604	
878	43.966457585	10.0.0.21	10.0.0.10	HTTP	404	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
1032	43.967649957	10.0.0.10	10.0.0.21	MP4	604	
1049	43.999434444	10.0.0.21	10.0.0.10	HTTP	405	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
1203	44.009438403	10.0.0.10	10.0.0.21	MP4	604	
1221	44.046298795	10.0.0.21	10.0.0.10	HTTP	405	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1
1375	44.047515218	10.0.0.10	10.0.0.21	MP4	604	
1392	44.062300611	10.0.0.21	10.0.0.10	HTTP	405	GET /videoB_640_400_1000k_dash.mp4 HTTP/1.1

Figura 17: Pedido de vídeo do cliente no portátil Bela

2.3 Questão 4

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

O arquivo DASH é conhecido como Media Presentation Description ou MPD. É baseado em XML e contém todas as informações necessárias para que o cliente descarregue e apresente um determinado conteúdo. Neste caso, o conteúdo multimédia é capturado e armazenado num servidor HTTP com o qual também é entregue.

Media Presentation Description descreve informações de segmento, tais como, o seu tempo, os seus endereços URL e as características de media, como a resolução de vídeo e taxas de bits. Pode ser organizada de diferentes maneiras, como SegmentList, SegmentTemplate, SegmentBase e SegmentTimeline, que contêm os fluxos de bits de multimédia reais na forma de chunks, em arquivos únicos ou múltiplos, dependendo do caso de uso.

O cliente DASH recupera e reproduz o conteúdo de vídeo usando as seguintes etapas:

- Em primeiro lugar, o cliente descarrega e lê o MPD para obter informações importantes, como locais de conteúdo, codificações de segmento, resolução, larguras de banda mínima e máxima, recursos de acessibilidade como legendas ocultas e restrições de conteúdo (DRM), localizações de componenetes de media na rede e outras características de conteúdo.

- Em segundo lugar, usando essas informações, o cliente DASH seleciona uma codificação de segmento apropriada e começa a transmitir o conteúdo por meio de solicitações HTTP GET.

- Por fim, o cliente armazena os dados em buffer à medida que são descarregados, ao mesmo tempo em que acompanha as flutuações na largura de banda da conexão. Se necessário, o cliente muda automaticamente para uma codificação de segmento diferente (com taxas de bits mais baixas ou mais altas) que seja mais compatível com a taxa de bits atual. Isso garante que o cliente mantenha um buffer suficiente em todo o vídeo sem descarregar mais dados do que é necessário.

A especificação MPEG-DASH define apenas o MPD e os formatos de segmento. A entrega do MPD e os formatos de codificação de media contendo os segmentos, bem como o comportamento do cliente para busca, heurística de adaptação e reprodução de conteúdo, estão fora do alcance do MPEG-DASH.

```
</SegmentList>
<Representation id="1" mimeType="video/mp4" codecs="avc3.64000c" width="160" height="100" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="87237">
 <BaseURL>videoB_160_100_200k_dash.mp4</BaseURL>
 <SegmentList timescale="15360" duration="7680">
   <SegmentURL mediaRange="927-5238" indexRange="927-970"/>
   <SegmentURL mediaRange="5239-8971" indexRange="5239-5282"/>
   <SegmentURL mediaRange="8972-13336" indexRange="8972-9015"/>
   <SegmentURL mediaRange="13337-23555" indexRange="13337-13380"/>
   <SegmentURL mediaRange="23556-29752" indexRange="23556-23599"/>
   <SegmentURL mediaRange="29753-35864" indexRange="29753-29796"/>
  <SegmentURL mediaRange="35865-36348" indexRange="35865-35908"/>
 </SegmentList>
```

Figura 18: Parte do ficheiro *MPD*

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

3.1 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.

O tipo de roteamento Multicast é um tipo de comunicação no qual um pacote é enviado para um grupo específico de dispositivos ou clientes, no caso específico usou-se um endereço de envio especial, que é um endereço de grupo, sendo este o 224.0.0.200 porta 5555. Os clientes da transmissão multicast devem ser membros de um grupo multicast lógico para receber as informações, isto é, em vez de ser enviado para um único destino como no Unicast, o tráfego mutlicast permite o envio de informações para um determinado grupo de clientes, cada um com um endereço diferente, ao mesmo tempo.

Numa transmissão Multicast, o servidor envia os pacotes apenas uma vez, ficando a cargo dos recetores captar esta transmissão e reproduzi-la. Esta técnica diminui consideravelmente o tráfego, como por exemplo vários clientes assistir a uma plataforma de streaming (Netfilx, Amazon prime, etc.).

Outra vantagem é a escalabilidade uma vez que permite que as aplicações sirvam um grande número de utilizadores sem sobrecarregar a rede.

Pelo contrário, a Unicast não permite escalabilidade aumentando linearmente com o número de clientes, no entanto, é um protocolo simples e eficaz quando se tem os recursos disponíveis.

Uma desvantagem em relação ao roteamento Unicast, é que o Multicast é mais complexo exigindo mais overhead devido à necessidade da gestão da árvore de distribuição, assim, cada vez que aumentamos o número de grupos e/ou utilizadores, é preciso mais controlo ao longo da adição. Além disso, o encaminhamento eficiente de pacotes não garante a entrega confiável dos dados uma vez que o controlo de erros, fluxo e congestionamento são tratados na camada superior.

No desenvolvimento desta etapa, ao utilizar transmissão Multicast, conseguimos verificar que, apesar de existir alterações de tamanho no protocolo, não há necessariamente um aumento ao longo do tempo.

4 Conclusão

A realização deste projeto possibilitou-nos, aplicar e expandir o nosso conhecimento sobre serviços de streaming de vídeo a pedido e em tempo real.

Conseguimos perceber também as opções a nível protocolar e conhecer melhor os formatos multimédia utilizados, bem como entender as diferenças entre a utilização do DASH e do HTTP simples.

Por fim permitiu-nos observar as diferenças entre roteamento Unicast e Multicast a nível de tráfego.

Concluindo, este trabalho ajudou-nos bastante a consolidar os conhecimentos sobre streaming de vídeo estudadas nas aulas práticas da unidade curricular de Engenharia de Serviços em Rede.