

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
MESTRADO EM ENGENHARIA INFORMÁTICA

Engenharia de Serviços em Rede - Trabalho Prático #1
Streaming de áudio e vídeo a pedido e em tempo real
Ano Letivo 2022/2023 - PL72

Simão Cunha (a93262) Tiago Silva (a93277)
Gonçalo Pereira (a93168)

26 de janeiro de 2023

Conteúdo

1	Questões	3
1.1	Capture três pequenas amostras de tráfego no link de saída do servidor, respetivamente com 1 cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay). Identifique a taxa em bps necessária (usando o ffmpeg -i videoA.mp4 e/ou o próprio wireshark), o encapsulamento usado e o número total de fluxos gerados. Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício (ex: capturas de ecrã)	3
1.2	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.	6
1.3	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.	7
1.4	Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.	9
1.5	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. . . .	9
2	Conclusão	12

1 Questões

- 1.1 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 ffplay). Identifique a taxa em bps necessária (usando o `ffmpeg -i videoA.mp4` e/ou o próprio wireshark), o encapsulamento usado e o número total de fluxos gerados. Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício (ex: capturas de ecrã)

O primeiro passo para a resolução deste exercício é a criação da topologia pedida. Assim, a mesma surgiu, tal como se segue abaixo:

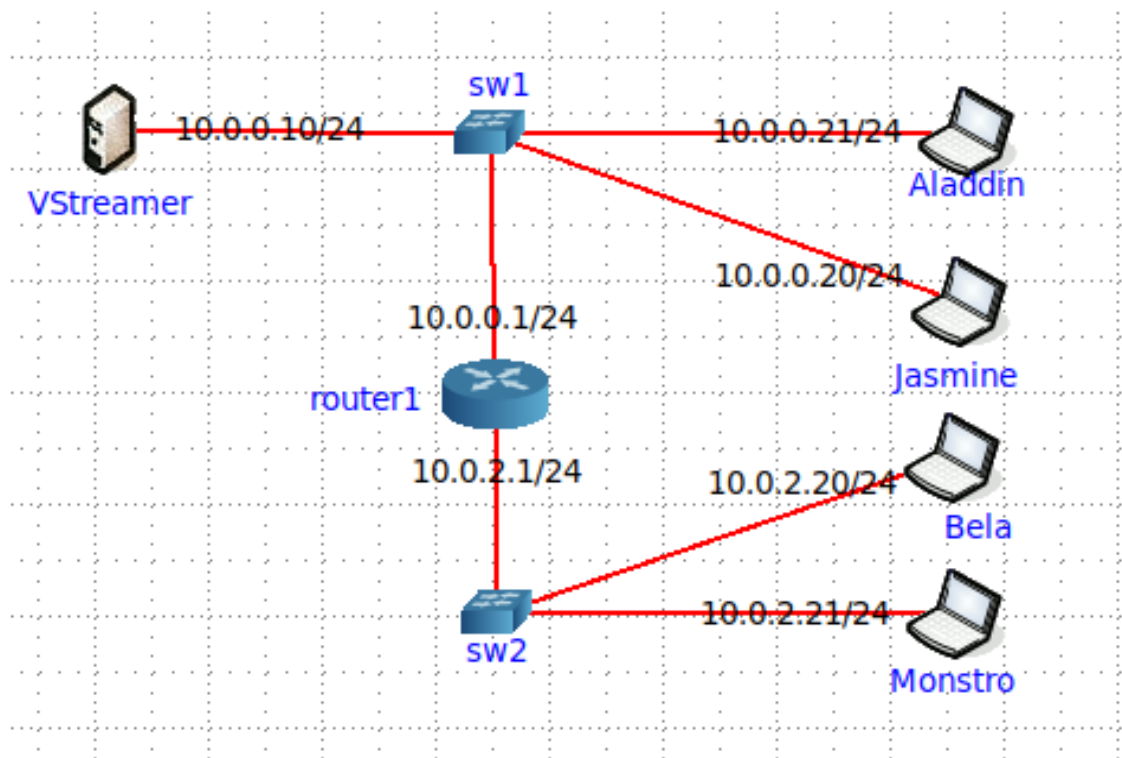


Figura 1: Topologia base

De seguida, de forma a verificar se a topologia foi criada corretamente, testamos a conectividade entre os diversos nós com o recurso ao comando `ping` e `traceroute`:

```

root@Monstro:/tmp/pycore.34747/Monstro.conf# ping 10.0.0.21
PING 10.0.0.21 (10.0.0.21) 56(84) bytes of data.
64 bytes from 10.0.0.21: icmp_seq=1 ttl=63 time=0.099 ms
64 bytes from 10.0.0.21: icmp_seq=2 ttl=63 time=0.117 ms
64 bytes from 10.0.0.21: icmp_seq=3 ttl=63 time=0.116 ms
64 bytes from 10.0.0.21: icmp_seq=4 ttl=63 time=0.116 ms

```

Figura 2: Ping do PC Monstro para o PC Aladdin

```

root@Jasmine:/tmp/pycore.34747/Jasmine.conf# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=63 time=0.095 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=63 time=0.120 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=63 time=0.118 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=63 time=0.121 ms
64 bytes from 10.0.2.20: icmp_seq=5 ttl=63 time=0.118 ms

```

Figura 3: Ping do PC Jasmine para o PC Bela

```

root@VStreamer:/tmp/pycore.34747/VStreamer.conf# traceroute 10.0.2.20
traceroute to 10.0.2.20 (10.0.2.20), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.048 ms  0.005 ms  0.003 ms
 2  10.0.2.20 (10.0.2.20)  0.014 ms  0.005 ms  0.005 ms

```

Figura 4: Traceroute do Server VStreamer para o PC Bela

Depois, passamos às capturas de tráfego pedidas ao longo desta etapa para as seguintes situações:

1. VStreamer a *streamar* no VLC e Jasmine a assistir no VLC;
2. VStreamer a *streamar* no VLC, Jasmine a assistir no VLC e Bela a assistir no Firefox;
3. VStreamer a *streamar* no VLC, Jasmine a assistir no VLC, Bela a assistir no Firefox e Monstro a assistir no *ffplay*;

Logo que efetuámos as capturas, abrimos as mesmas com recurso ao Wireshark (*Statistics* → *Conversations* → *IPv4*), onde descobrimos a taxa em bps do tráfego do *link* de saída do servidor para os respetivos clientes:

Ethernet · 3		IPv4 · 2		IPv6 · 1	TCP · 1	UDP						
Address A	Address 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	
10.0.0.1	224.0.0.5	5	390	5	390	0	0	0.634587	8.0495		387	
10.0.0.10	10.0.0.20	212	143 k	106	136 k	106	6996	0.000000	9.6811		113 k	

Figura 5: Wireshark da situação 1 (113 K)

Ethernet · 4	IPv4 · 3	IPv6 · 1	TCP · 2	UDP							
Address A	Address 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
10.0.0.1	224.0.0.5	4	312	4	312	0	0	1.051446	6.0019	415	
10.0.0.10	10.0.0.20	172	117 k	86	112 k	86	5676	0.000000	7.6983	116 k	
10.0.0.10	10.0.2.20	172	117 k	86	112 k	86	5676	0.000141	7.6983	116 k	

Figura 6: Wireshark da situação 2 (116 K + 116 K)

Ethernet · 4	IPv4 · 4	IPv6 · 1	TCP · 3	UDP							
Address A	Address 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
10.0.0.1	224.0.0.5	4	312	4	312	0	0	0.471365	6.0258	414	
10.0.0.10	10.0.0.20	176	118 k	88	112 k	88	5808	0.000000	7.6930	117 k	
10.0.0.10	10.0.2.20	176	118 k	88	112 k	88	5808	0.007313	7.6858	117 k	
10.0.0.10	10.0.2.21	176	118 k	88	112 k	88	5808	0.012238	7.6809	117 k	

Figura 7: Wireshark da situação 3 (117 K + 117 K + 117 K)

Nota: Ao afirmarmos que o tráfego é do tipo $a + b + c$, queremos referir-mo-nos à ligação do VStreamer para a Jasmine de a bps, para a Bela de b bps e para o Monstro de c bps.

O próximo passo é descobrir qual o encapsulamento usado em cada uma das situações. Conforme a imagem abaixo, que se refere à situação 3 (VStreamer a streamar no VLC, Jasmine a assistir no VLC, Bela a assistir no Firefox e Monstro a assistir no ffmpeg), podemos identificar que os protocolos usados são - os mesmos são aplicados também às situações 1 e 2:

- Ethernet (Nível 2 - lógico - do modelo OSI);
- IP (Nível 3 - rede - do modelo OSI);
- TCP (Nível 4 - transporte - do modelo OSI);
- HTTP (Nível 7 - aplicação - do modelo OSI) [usado apenas no pedido HTTP GET - não será necessário posteriormente porque irá estabelecer-se a comunicação por TCP];

1	0.000000000	10.0.0.10	10.0.0.20	TCP	1514	8080 → 58160 [ACK] Seq=1 Ack=1 Win=509 Len=1448 TSval=4259825
Wireshark · Packet 1 · videoAStream_3.pcapng						
▶ Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth8.0.e6, id 0 ▶ Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01) ▶ Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20 ▶ Transmission Control Protocol, Src Port: 8080, Dst Port: 58160, Seq: 1, Ack: 1, Len: 1448 ▶ Hypertext Transfer Protocol						

Figura 8: Encapsulamento VStreamer → Jasmine

7	0.007312603	10.0.0.10	10.0.2.20	TCP	1514	8080 → 49736 [ACK] Seq=1 Ack=1 Win=507 Len=1448 TSval=4092933
Wireshark · Packet 7 · videoAStream_3.pcapng						
▶ Frame 7: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth8.0.e6, id 0 ▶ Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:03 (00:00:00:aa:00:03) ▶ Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.2.20 ▶ Transmission Control Protocol, Src Port: 8080, Dst Port: 49736, Seq: 1, Ack: 1, Len: 1448 ▶ Hypertext Transfer Protocol						

Figura 9: Encapsulamento VStreamer → Bela

13	0.012238109	10.0.0.10	10.0.2.21	TCP	1514	8080 → 55556 [ACK] Seq=1 Ack=1 Win=509 Len=1448 TSval=3857697...
Wireshark - Packet 13 - videoASream_3.pcapng						
▶ Frame 13: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface veth8.0.e6, id 0						
▶ Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:03 (00:00:00:aa:00:03)						
▶ Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.2.21						
▶ Transmission Control Protocol, Src Port: 8080, Dst Port: 55556, Seq: 1, Ack: 1, Len: 1448						
▶ Hypertext Transfer Protocol						

Figura 10: Encapsulamento VStreamer → Monstro

De forma a facilitar a consulta dos resultados, elaboramos a seguinte tabela:

	taxa em bps	encapsulamento usado	nº total de fluxos gerados
1 cliente (VLC)	113 K	ETH,IP,TCP,HTTP	1
2 clientes (VLC e Firefox)	116 K + 116 K	ETH,IP,TCP,HTTP	2
3 clientes (VLC, Firefox e ffmpeg)	117 K + 117 K + 117 K	ETH,IP,TCP,HTTP	3

Nota: Nº total de fluxos gerados = nº de clientes na ligação

De forma a entendermos a escalabilidade desta solução de *streaming*, podemos observar a taxa do *link* de saída do servidor com os diferentes números de clientes. Verificamos que esta mesma taxa tende a aumentar quantos mais clientes existirem conectados, implicando um maior envio de pacotes dos dados aos mesmos. Isto leva-nos a concluir que esta solução tende a ser incapaz de responder a todos os pedidos com um aumento astronómico do número de entidades a assistir à *stream*, sendo esta solução de *streaming* pouco escalável.

1.2 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.

De forma a calcular a largura de banda necessária em *bps* para que o cliente de *streaming* consiga receber o vídeo no **Firefox**, recorremos ao **Wireshark** (Statistics → Conversations) e observamos que a taxa de *bps* do *streaming* era 1462 Kbps, logo concluímos que uma largura de banda superior a 1462 Kbps seja a necessária.

Ethernet · 3	IPv4 · 2	IPv6 · 1	TCP · 5	UDP							
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	
10.0.0.1	224.0.0.5	10	780	10	780	0	0	0.000000	18.0227	346	
10.0.0.10	10.0.2.20	444	449 k	307	438 k	137	10 k	15.274635	2.3985	1462 k	

Figura 11: Taxa de bps do *streaming* do vídeo no Firefox

A pilha protocolar aqui presente é, tal como se comprova na aba *Protocol Hierarchy* do Wireshark:

- Ethernet (camada de ligação);
- IP (camada de rede);
- TCP (camada de transporte);
- HTTP (camada aplicacional).

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	456	100.0	450453	199 k	0	0	0
▼ Ethernet	100.0	456	1.4	6384	2833	0	0	0
▼ Internet Protocol Version 6	0.4	2	0.0	80	35	0	0	0
Open Shortest Path First	0.4	2	0.0	72	31	2	72	31
▼ Internet Protocol Version 4	99.6	454	2.0	9080	4030	0	0	0
▼ Transmission Control Protocol	97.4	444	96.4	434397	192 k	434	429544	190 k
▼ Hypertext Transfer Protocol	2.2	10	93.3	420109	186 k	8	2430	1078
MP4 / ISOBMFF file format	0.2	1	92.5	416799	185 k	1	417004	185 k
Line-based text data	0.2	1	0.1	487	216	1	487	216
Open Shortest Path First	2.2	10	0.1	440	195	10	440	195

Figura 12: Pilha protocolar

1.3 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.

R:

Para este exercício, consultamos o ficheiro `video_manifest.mpd` para sabermos valores médios dos *bitrates* necessários para *streamar* cada tipo de vídeo. Ora, observando o seu conteúdo, temos:

```

1  ...
2  <Representation id="1" mimeType="video/mp4" codecs="avc3.64000c"
3  width="160" height="100" frameRate="30" sar="1:1"
4  startWithSAP="0" bandwidth="104914">
5  ...
6  <Representation id="2" mimeType="video/mp4" codecs="avc3.640014"
7  width="320" height="200" frameRate="30" sar="1:1"
8  startWithSAP="0" bandwidth="240460">
9  ...
10 <Representation id="3" mimeType="video/mp4" codecs="avc3.64001e"
11 width="640" height="400" frameRate="30" sar="1:1"
12 startWithSAP="0" bandwidth="605350">
13 ...

```

Analisando este mesmo ficheiro, podemos retirar algumas conclusões:

- O vídeo tem três resoluções possíveis;
- Se a largura de banda estiver entre 104914 e 240460, o vídeo sairá com a pior resolução possível;
- Se a largura de banda estiver entre 240460 e 605350, o vídeo sairá com resolução média;
- Se a largura de banda estiver superior a 605350, o vídeo sairá com a melhor resolução possível;
- De notar, que estes valores para as larguras de banda são apenas guias teóricos, uma vez que estes não têm em conta o *overhead* protocolar, logo, na prática, teria que se trabalhar com valores um pouco mais elevados;

Assim, alterando devidamente o débito na topologia - os valores foram escolhidos aleatoriamente desde que cumprissem os intervalos acima descritos - temos:

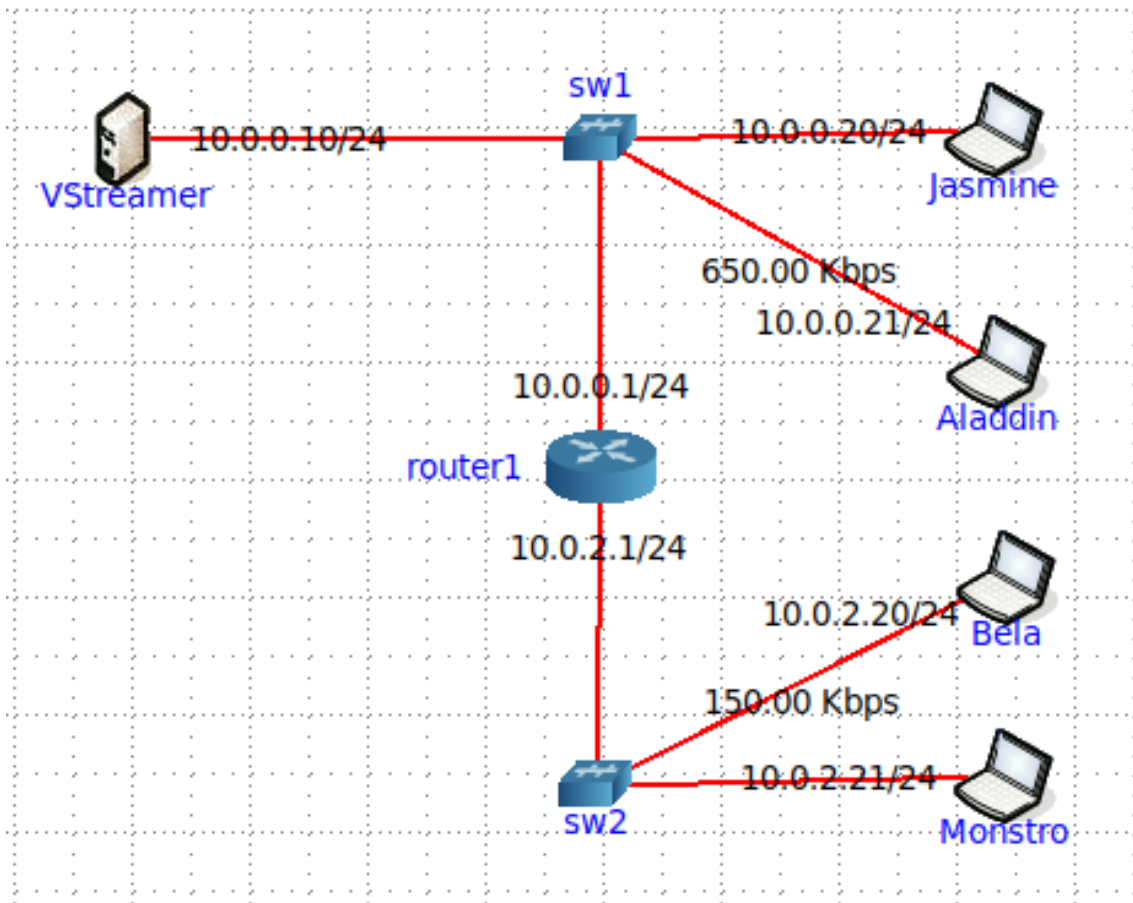


Figura 13: Topologia com as devidas alterações nos débitos

Logo, segue abaixo uma imagem que reflete a Bela a assistir ao vídeo com pior resolução e o Aladdin e assistir ao vídeo com melhor resolução.

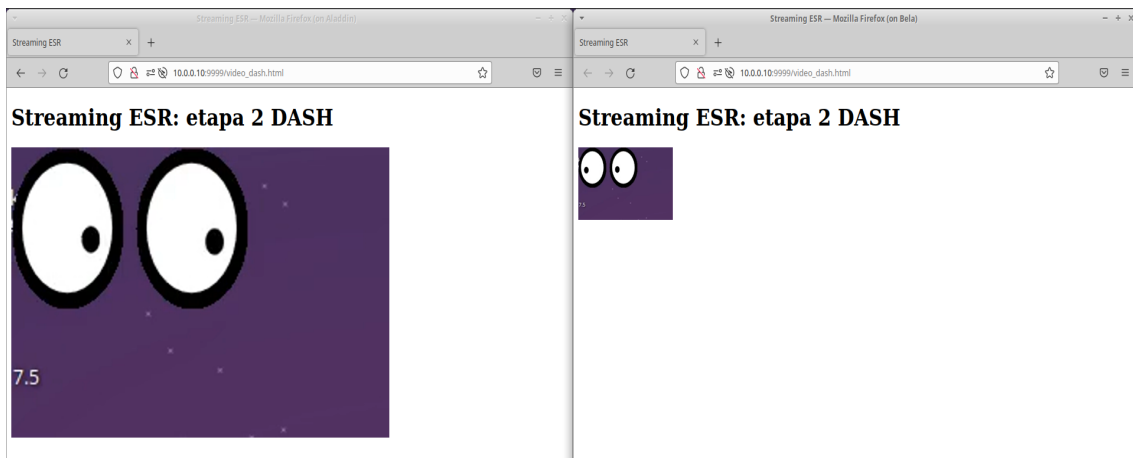


Figura 14: Aladdin e Bela a verem as respectivas *streams* (respetivamente)

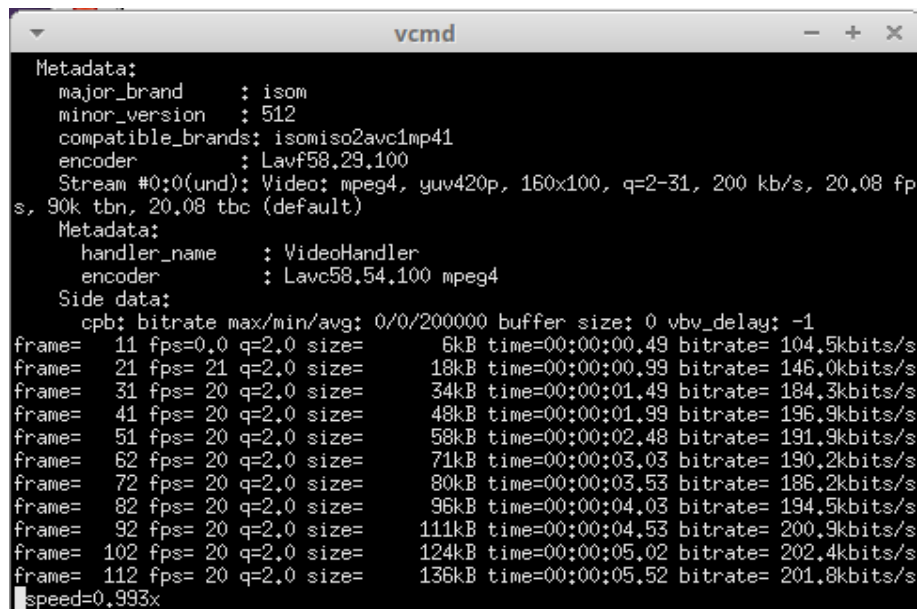
1.4 Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.

DASH (*Dynamic Adaptive Streaming over HTTP*), tal como o nome indica, permite o *streaming* com débitos adaptativos.

Inicialmente são criadas versões do mesmo vídeo com diferentes dimensões. De seguida, é gerado o ficheiro MPD que descreve as diferentes versões dos vídeos bem como informações sobre a sua localização, resolução e largura de banda mínima e máxima. A primeira coisa que o cliente faz é pedir o ficheiro MPD. No entanto, não foi necessário isso acontecer porque o sistema de ficheiros é partilhado, tendo já acesso ao ficheiro pretendido. A qualquer momento, dependendo da largura de banda disponível, o cliente pode pedir um *chunk* com mais ou menos qualidade, fazendo uso do *manifest file* (MPD).

1.5 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.

De forma a resolvermos este exercício, utilizámos a topologia da Etapa 1 e iniciamos uma sessão de *streaming* com RTP no VStreamer através do comando *ffmpeg* para o portátil Monstro.



```
vcmd
Metadata:
  major_brand      : isom
  minor_version    : 512
  compatible_brands: isomiso2avc1mp41
  encoder          : Lavf58.29.100
  Stream #0:0(und): Video: mpeg4, yuv420p, 160x100, q=2-31, 200 kb/s, 20.08 fps, 90k tbn, 20.08 tbc (default)
Metadata:
  handler_name     : VideoHandler
  encoder          : Lavc58.54.100 mpeg4
Side data:
  cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: -1
frame= 11 fps=0.0 q=2.0 size=      6kB time=00:00:00.49 bitrate= 104.5kbits/s
frame= 21 fps= 21 q=2.0 size=     18kB time=00:00:00.99 bitrate= 146.0kbits/s
frame= 31 fps= 20 q=2.0 size=     34kB time=00:00:01.49 bitrate= 184.3kbits/s
frame= 41 fps= 20 q=2.0 size=     48kB time=00:00:01.99 bitrate= 196.9kbits/s
frame= 51 fps= 20 q=2.0 size=     58kB time=00:00:02.48 bitrate= 191.9kbits/s
frame= 62 fps= 20 q=2.0 size=     71kB time=00:00:03.03 bitrate= 190.2kbits/s
frame= 72 fps= 20 q=2.0 size=     80kB time=00:00:03.53 bitrate= 186.2kbits/s
frame= 82 fps= 20 q=2.0 size=     96kB time=00:00:04.03 bitrate= 194.5kbits/s
frame= 92 fps= 20 q=2.0 size=    111kB time=00:00:04.53 bitrate= 200.9kbits/s
frame= 102 fps= 20 q=2.0 size=    124kB time=00:00:05.02 bitrate= 202.4kbits/s
frame= 112 fps= 20 q=2.0 size=    136kB time=00:00:05.52 bitrate= 201.8kbits/s
speed=0.993x
```

Figura 15: VStreamer streaming com RTP com o *ffmpeg* para o Monstro

De seguida, iniciamos um cliente *ffplay* no portátil Monstro e capturámos o tráfego com recurso ao *Wireshark* no *link* de saída do VStreamer.

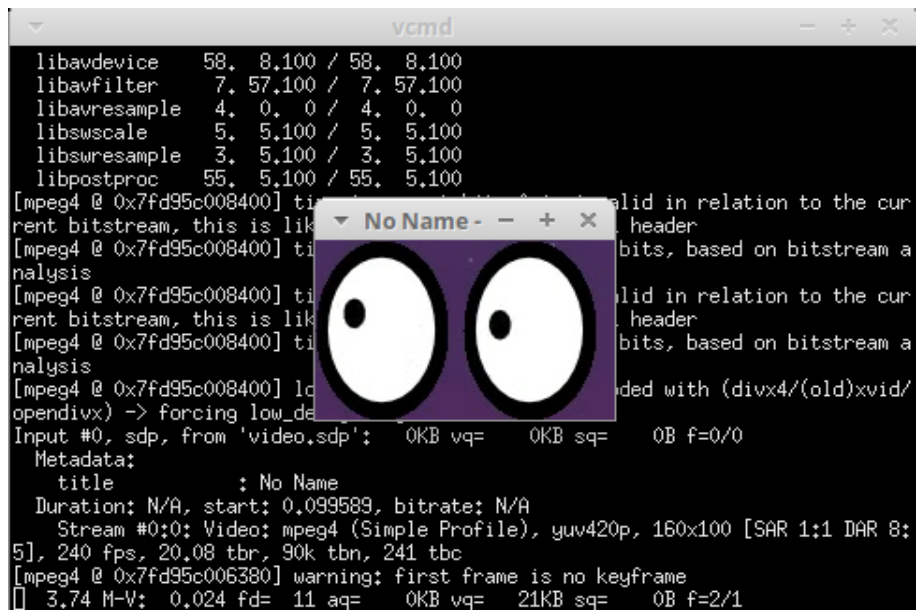


Figura 16: Monstro ligado ao cliente ffplay

De forma a testarmos o *multicast*, criámos uma nova topologia com apenas um *switch* com o servidor e quatro portáteis ligados a esse *switch*.

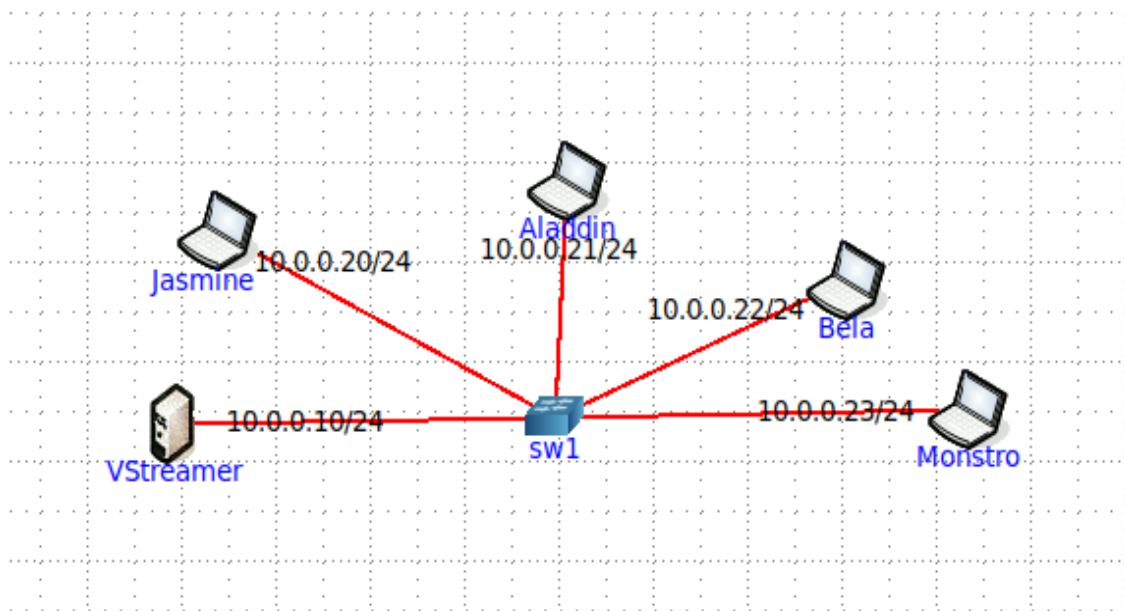


Figura 17: Nova topologia para testar o multicast

Depois, iniciamos uma sessão de *streaming multicast* no VStreamer através do comando `ffmpeg`.

```

vcmd
minor_version : 512
compatible_brands: isomiso2avc1mp41
Duration: 00:00:10.11, start: 0.000000, bitrate: 18 kb/s
Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 160x100,
16 kb/s, 20.08 fps, 20.08 tbr, 15424 tbn, 40.17 tbc (default)
Metadata:
  handler_name : VideoHandler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> mpeg4 (native))
Press [q] to stop, [?] for help
Output #0, sap, to 'sap://224.0.0.200:5555':
Metadata:
  major_brand : isom
  minor_version : 512
  compatible_brands: isomiso2avc1mp41
  encoder : Lavf58.29.100
Stream #0:0(und): Video: mpeg4, yuv420p, 160x100, q=2-31, 200 kb/s, 20.08 fp
s, 90k tbn, 20.08 tbc (default)
Metadata:
  handler_name : VideoHandler
  encoder : Lavc58.54.100 mpeg4
Side data:
  cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: -1
frame= 82 fps= 20 q=2.0 size=N/A time=00:00:04.03 bitrate=N/A speed=0.998x

```

Figura 18: VStreamer streaming multicast com o ffmpeg

De seguida, iniciamos um cliente `ffplay` em cada um dos portáteis e capturámos o tráfego com recurso ao `Wireshark` no *link* de saída do VStreamer.

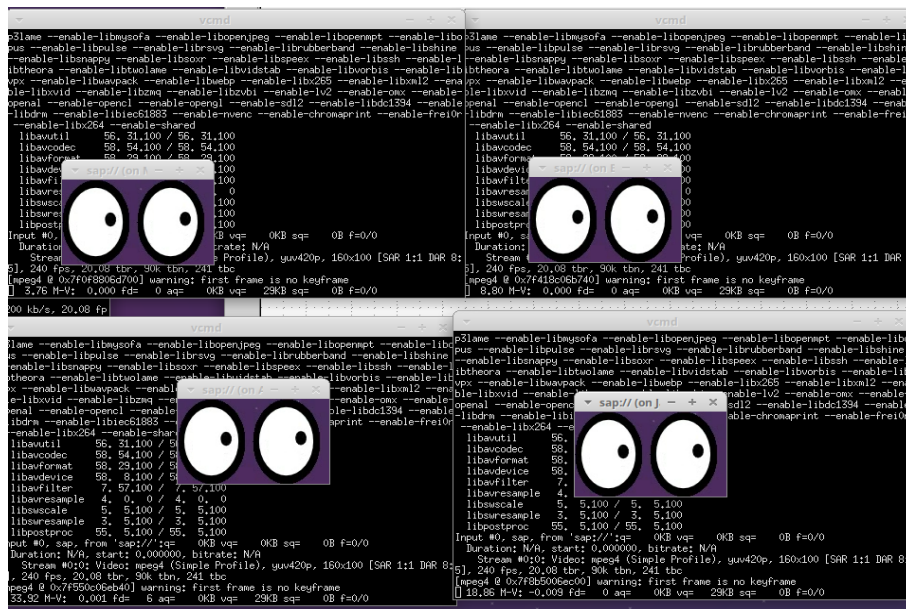


Figura 19: Portáteis (Jasmine, Aladdin, Bela e Monstro) com sessão de cliente iniciada

Analisando as capturas de tráfego, chegamos às seguintes conclusões:

- Na situação *unicast*, o servidor inicia a comunicação enviando uma cópia para o cliente com que comunica;
- No cenário *multicast*, o servidor difunde uma única cópia dos dados para um grupo de clientes;

- **Multicast** apresenta grandes vantagens quanto à escalabilidade, tendo em conta que, por muito que o número de clientes aumente, a cópia dos dados circula pelos grupos de clientes;
- No caso do **unicast**, com o aumento do número de clientes, teria de se estabelecer comunicação e enviar cópias de dados para cada um dos clientes;
- O **multicast** apresenta desvantagens ao nível do tráfego de rede, visto que, numa abordagem *flood* e *prune* do **multicast**, propaga uma cópia dos dados pela rede. No entanto, os clientes não interessados nesses mesmos dados não os irão processar, acabando por este facto não ter muito impacto;
- A maior desvantagem no **multicast** seria o facto deste não ser **plug and play**, ou seja, é preciso primeiro implementar a estrutura para **multicast**.

2 Conclusão

Com a resolução deste primeiro trabalho prático da UC de Engenharia de Serviços em Rede, cremos que atingimos os objetivos definidos pela equipa docente para este TP1.

Abordamos variados temas como *streaming* apenas por HTTP sem adaptação do débito; *streaming* por HTTP com adaptação do débito com recurso ao **ffmpeg** e ao **DASH**; RTP/RTCP em *unicast* e anúncios SAP com 4 clientes mudando de *unicast* para *multicast* (ao nível da rede); escalabilidade dos tipos de *streaming*.

Acreditamos que estes mesmos temas ajudar-nos-ão a entender melhor como funcionam os diferentes tipos *streaming* e outros conceitos subjacentes como, por exemplo, a camada aplicacional do modelo OSI.