

UNIVERSIDADE DO MINHO

MESTRADO EM ENGENHARIA INFORMÁTICA

---

Engenharia de Serviços em Rede

**Grupo 73**

---

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

Catarina Morales (PG50289)

Diogo Casal Novo (PG50342)

Rui Armada (PG50737)

# 1 Questão 1

1.1 Capture três pequenas amostras de trágefo 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).

Primeiro foi necessário utilizar o `ffmpeg` de forma a observar qual é a taxa de *bps* necessário para dar *upload* e *download* `videoA.mp4` e observar o seu encapsulamento.

```
core@ubuncore:~$ ffmpeg -i videoA.mp4
ffmpeg version 4.2.7-0ubuntu1.3 Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-1ubuntu0.1)
configuration: --prefix=/usr --extra-version=0ubuntu0.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl --enable-stripping --enable-avresample --disable-filter=resample --enable-avisynth --enable-ladspa --enable-libaom --enable-libbluray --enable-libbsz --enable-libcaca --enable-libdcod --enable-libfdk-aac --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-liblame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwpack --enable-libwebp --enable-libx264 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-libzv2 --enable-omx --enable-opencn --enable-opengl --enable-estd --enable-libmp3lame --enable-libopencore-amrnm --enable-libopencore-amrnb --enable-libopusfile --enable-libsvtav1 --enable-libvpx --enable-libx265 --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
libavproc     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: isomiso2avclmp41
    encoder : Lavf58.29.100
Duration: 00:00:19.65, start: 0.000000, bitrate: 10 kb/s
  Stream #0:0[und]: Video: h264 (high) (avc1 / 0x31637661), yuv420p, 160x100, 7 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
  Metadata:
    handler_name : VideoHandler
At least one output file must be specified
core@ubuncore:~$
```

Figura 1: `ffmpeg -i videoA.mp4`

Pimeiramente pode ser observado na figura 1 que o video necessita de uma taxa de *bps* teórica de  $7\text{ Kb/s}$  para ser processado.<sup>1</sup> Porém, ao analisar o *wireshark* pode-se obeservar que com apenas 1 cliente (VLC aberto no PC Jasmine) a taxa de *bps* é de  $125\text{ Kb/s}$ .

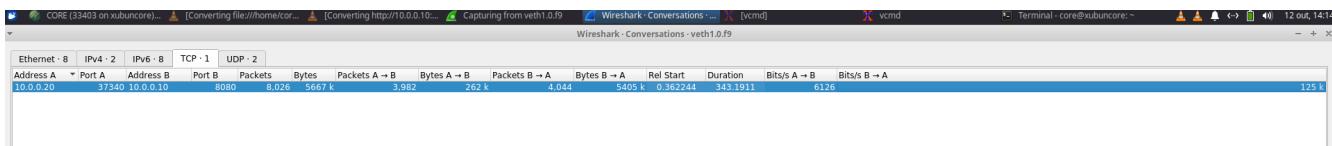


Figura 2: Taxa de *bps* real

No que toca ao encapsulamento pode-se observar na figura 1 que o ficheiro video recorre a vários tipos de encapsulamento. Os tipos de encapsulamento observados são:

- mov
- m4a
- 3g2
- mp4
- 3gp
- mj2

<sup>1</sup>Taxa de *bps* - forma de medir a velocidade de processamento um ficheiro

Também se pode constatar que o encapsulamento do ficheiro decorre em todos os diversos níveis da camada protocolar: camada de transporte (**TCP**), camada de rede (**IPv4**), camada de aplicação (**HTTP**) e na camada de ligação de dados.

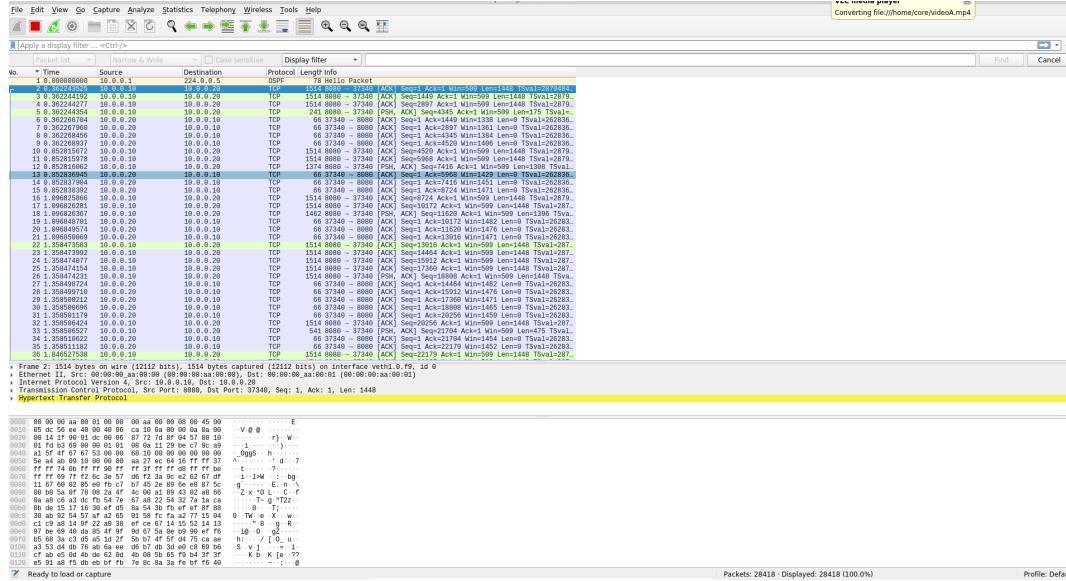


Figura 3: Encapsulamento TCP com 1 cliente (VLC)

Relativamente a capturas com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay) o encapsulamento é o mesmo, tal pode ser observado nas seguintes figuras.

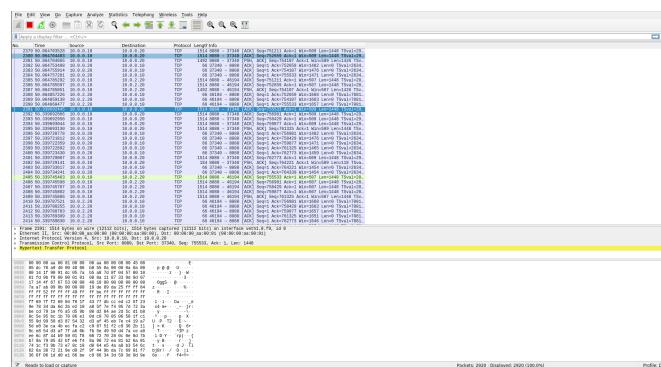


Figura 4: Encapsulamento TCP com 2 clientes

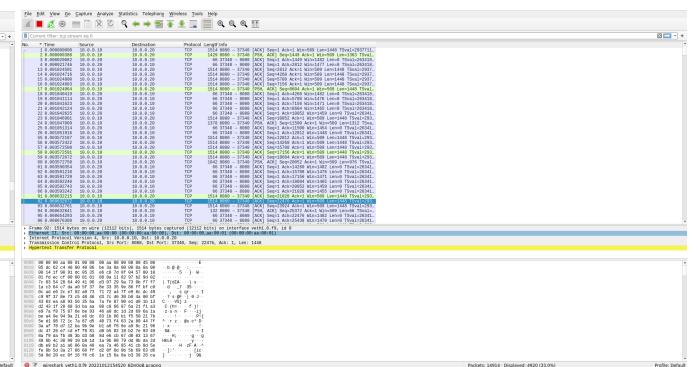


Figura 5: Encapsulamento TCP com 3 clientes

Para o efeito de comprovar o número de fluxos gerados é utilizado o filtro ***tcp.stream eq*** para verificar a existencia de diferentes *streams*. Na primeira captura onde apenas existe um cliente (VLC) pode-se observar que apenas ocorre 1 único fluxo, como se pode observar na figura 6. Após a utilização do filtro indicado pode-se observar que apenas existe a *stream 0* podendo-se afirmar que apenas existe um fluxo.

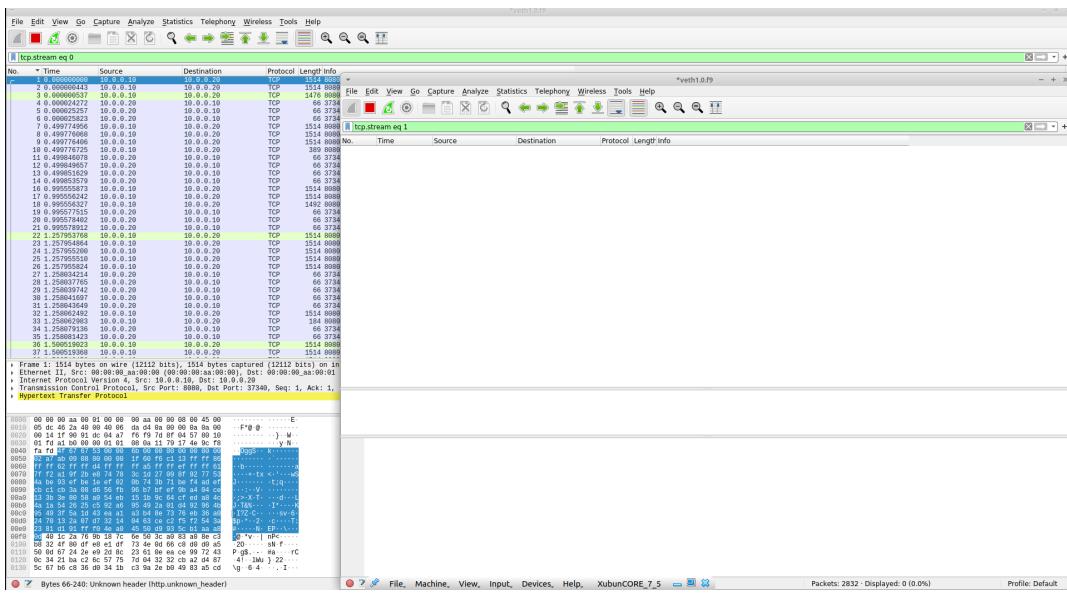


Figura 6: Fluxos gerados com 1 cliente

Na captura com 2 clientes (VLC e Firefox) pode-se observar a existência de 2 fluxos. Utilizando novamente o filtro ***tcp.stream eq*** pode-se observar a existência da *stream 0* e da *stream 2*, sendo possível afirmar que existem 2 fluxos.

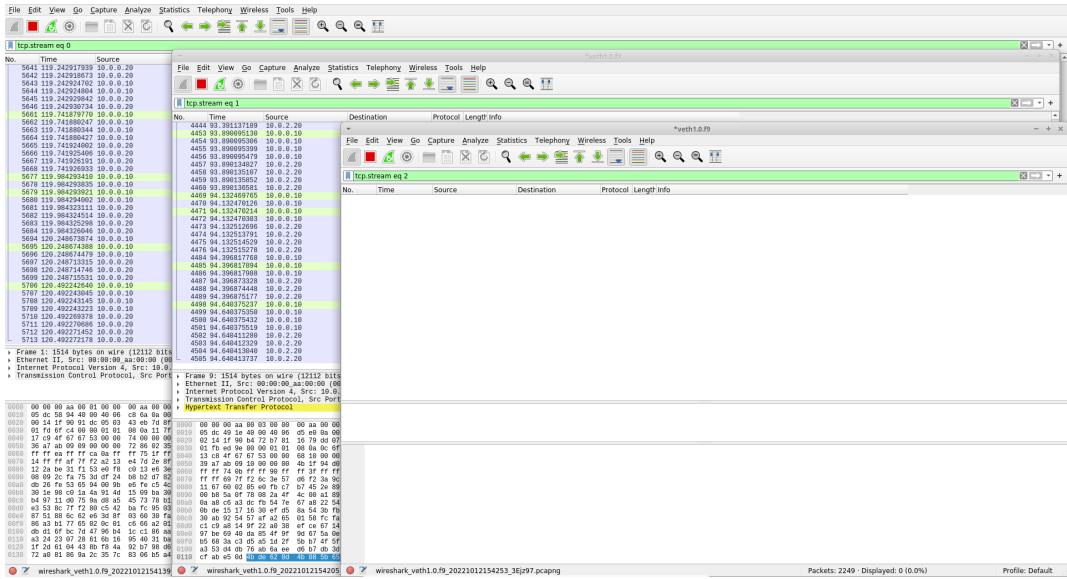


Figura 7: Fluxos gerados com 2 clientes

Na última captura, a captura com 3 clientes (VLC, Firefox e *ffplay*) pode-se observar a existência de 3 fluxos após a utilização do filtro já mencionado anteriormente. Pode-se observar a existência da *stream 0*, *stream 1* e *stream 2* concluindo-se que existem 3 fluxos.

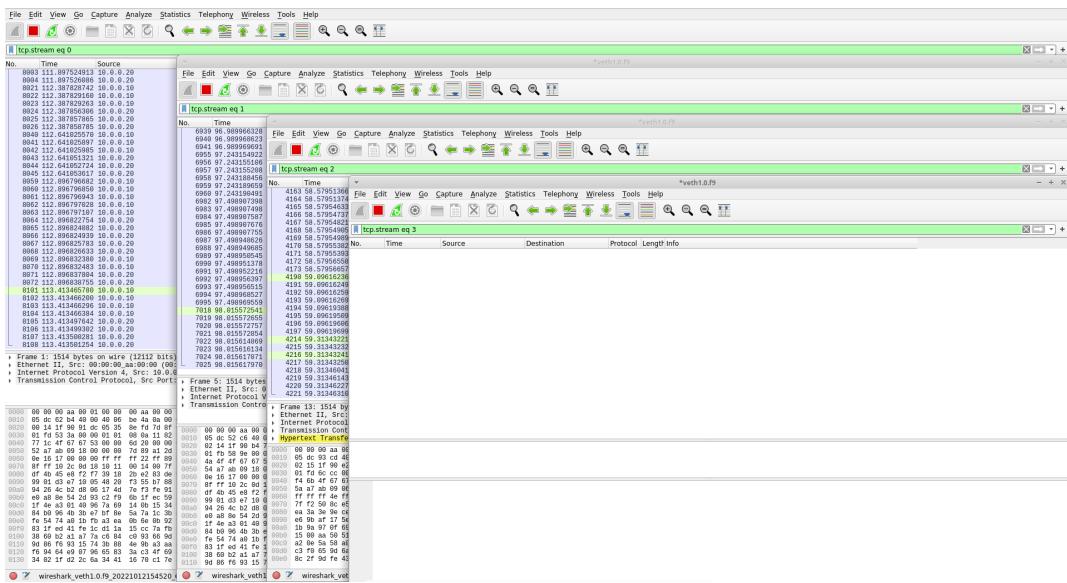


Figura 8: Fluxos gerados com 3 clientes

Pela observação das figuras apresentadas anteriormente podemos verificar que o servidor envia a resposta de pedidos individualmente para cada cliente pelo que se pode concluir que esta não é uma solução muito escalável.

## 2 Questão 2

2.1 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 fim de ser possível descobrir a largura de banda necessária para que o cliente de streaming consiga receber o vídeo no firefox consulta-mos o ficheiro **video\_manifest.mpd**. A largura de banda necessaria para o video 160\*100 é de 73030, para o video de 320\*200 é de 163532 e por fim, para o video de 640\*400 é de 409160.

```
<File id="file1">
<adaptationSet segmentDuration="1000000000">
<segmentList id="list1">
<Initialization sourceURL="video.manifest.init.mp4"/>
<SegmentList id="list1">
<segment id="seg1" type="mp4-video.m4v" codec="avc3.640000" width="160" height="100" frameRate="30000/1001" par="8/5" lang="und">
<segmentURL>vodvod_160_700_dash.m4v</segmentURL>
<segmentRRL mediRange="0-1000000000" indexRange="0-1000000000" mediaRange="0-1000000000" mediaRange="0-1000000000"/>
<segmentRRL mediRange="926-5279" indexRange="926-5279" mediaRange="926-969" mediaRange="926-969"/>
<segmentRRL mediRange="5271-8959" indexRange="5271-5314" mediaRange="5271-933" mediaRange="5271-933"/>
<segmentRRL mediRange="8960-16060" indexRange="8960-16060" mediaRange="8960-16060" mediaRange="8960-16060"/>
<segmentRRL mediRange="10956-15660" indexRange="10956-16060" mediaRange="10956-16060" mediaRange="10956-16060"/>
<segmentRRL mediRange="15661-16288" indexRange="15661-16288" mediaRange="15661-16288" mediaRange="15661-16288"/>
<segmentRRL mediRange="16888-23890" indexRange="16888-16911" mediaRange="16888-16911" mediaRange="16888-16911"/>
<segmentRRL mediRange="23891-24121" indexRange="23891-24121" mediaRange="23891-24121" mediaRange="23891-24121"/>
<segmentRRL mediRange="22980-23551" indexRange="22980-23551" mediaRange="22980-23551" mediaRange="22980-23551"/>
<segmentRRL mediRange="23552-24121" indexRange="23552-23995" mediaRange="23552-23995" mediaRange="23552-23995"/>
<segmentRRL mediRange="23996-24121" indexRange="23996-24121" mediaRange="23996-24121" mediaRange="23996-24121"/>
<segmentRRL mediRange="30871-30884" indexRange="30871-30114" mediaRange="30871-30884" mediaRange="30871-30884"/>
<segmentRRL mediRange="30885-30898" indexRange="30885-30898" mediaRange="30885-30898" mediaRange="30885-30898"/>
<segmentRRL mediRange="34441-34444" indexRange="34441-34444" mediaRange="34441-34444" mediaRange="34441-34444"/>
<segmentRRL mediRange="43983-52678" indexRange="43983-44006" mediaRange="43983-52678" mediaRange="43983-52678"/>
<segmentRRL mediRange="68656-72556" indexRange="68656-68699" mediaRange="68656-72556" mediaRange="68656-68699"/>
<segmentRRL mediRange="72557-72573" indexRange="72557-72573" mediaRange="72557-72573" mediaRange="72557-72573"/>
<segmentRRL mediRange="7295-8149" indexRange="7295-72548" mediaRange="7295-8149" mediaRange="7295-8149"/>
<segmentRRL mediRange="8150-8234" indexRange="8150-8153" mediaRange="8150-8234" mediaRange="8150-8153"/>
<segmentRRL mediRange="89751-92634" indexRange="89751-89794" mediaRange="89751-92634" mediaRange="89751-89794"/>
<segmentRRL mediRange="92635-91129" indexRange="92635-92978" mediaRange="92635-91129" mediaRange="92635-92978"/>
<segmentRRL mediRange="99131-99697" indexRange="99131-99172" mediaRange="99131-99697" mediaRange="99131-99172"/>
</SegmentList>
</adaptationSet>
</File>
```

Figura 9: Video 160\*100

```

<Representation id="avc3_640x144" codectype="avc3_640x144" width="320" height="280" framerate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="163532">
  <SegmentList timescale="30000" duration="15000">
    <SegmentRl indexRange="0-295" mediaRange="0-9657" />
    <SegmentRl indexRange="18596-17354" mediaRange="18596-16339" />
    <SegmentRl indexRange="17355-20358" mediaRange="17355-17278" />
    <SegmentRl indexRange="20359-23352" mediaRange="20359-19908" />
    <SegmentRl indexRange="23353-26345" mediaRange="23353-22273" />
    <SegmentRl indexRange="26346-29338" mediaRange="26346-24257" />
    <SegmentRl indexRange="29339-32331" mediaRange="29339-27257" />
    <SegmentRl indexRange="32332-35324" mediaRange="32332-31257" />
    <SegmentRl indexRange="35325-38317" mediaRange="35325-34257" />
    <SegmentRl indexRange="38318-41310" mediaRange="38318-37257" />
    <SegmentRl indexRange="41311-44303" mediaRange="41311-36257" />
    <SegmentRl indexRange="44304-48295" mediaRange="44304-40883" />
    <SegmentRl indexRange="48296-52287" mediaRange="48296-44283" />
    <SegmentRl indexRange="52288-62197" mediaRange="52288-56248" />
    <SegmentRl indexRange="62198-62921" mediaRange="62198-62964" />
    <SegmentRl indexRange="62922-66485" mediaRange="62922-65233" />
    <SegmentRl indexRange="66486-94167" mediaRange="66486-94187" />
    <SegmentRl indexRange="94168-12397" mediaRange="94168-12399" />
    <SegmentRl indexRange="12398-15369" mediaRange="12398-15369" />
    <SegmentRl indexRange="15370-18399" mediaRange="15370-15399" />
    <SegmentRl indexRange="18400-162008" mediaRange="18400-162847" />
    <SegmentRl indexRange="162009-192028" mediaRange="162009-192847" />
    <SegmentRl indexRange="192029-202379" mediaRange="192029-193833" />
    <SegmentRl indexRange="202380-208028" mediaRange="202380-202422" />
    <SegmentRl indexRange="208029-70897" mediaRange="208029-208777" />
    <SegmentRl indexRange="70898-222529" mediaRange="70898-222529" />
  </SegmentList>
</Representation>
<Representation id="avc3_480x80" minTbr="100000" videoCodec="avc3_480x80" width="480" height="80" framerate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="480160">
  <SegmentList timescale="30000" duration="15000">
    <SegmentRl indexRange="0-295" mediaRange="0-9657" />
    <SegmentRl indexRange="18596-17354" mediaRange="18596-16339" />
    <SegmentRl indexRange="17355-20358" mediaRange="17355-17278" />
    <SegmentRl indexRange="20359-23352" mediaRange="20359-19908" />
    <SegmentRl indexRange="23353-26345" mediaRange="23353-22273" />
    <SegmentRl indexRange="26346-29338" mediaRange="26346-24257" />
    <SegmentRl indexRange="29339-32331" mediaRange="29339-22273" />
    <SegmentRl indexRange="32332-35324" mediaRange="32332-24257" />
    <SegmentRl indexRange="35325-38317" mediaRange="35325-22273" />
    <SegmentRl indexRange="38318-41310" mediaRange="38318-24257" />
    <SegmentRl indexRange="41311-44303" mediaRange="41311-24257" />
    <SegmentRl indexRange="44304-48295" mediaRange="44304-24257" />
    <SegmentRl indexRange="48296-52287" mediaRange="48296-24257" />
    <SegmentRl indexRange="52288-62197" mediaRange="52288-24257" />
    <SegmentRl indexRange="62198-62921" mediaRange="62198-24257" />
    <SegmentRl indexRange="62922-66485" mediaRange="62922-24257" />
    <SegmentRl indexRange="66486-94167" mediaRange="66486-24257" />
    <SegmentRl indexRange="94168-12397" mediaRange="94168-24257" />
    <SegmentRl indexRange="12398-15369" mediaRange="12398-24257" />
    <SegmentRl indexRange="15370-18399" mediaRange="15370-24257" />
    <SegmentRl indexRange="18400-162008" mediaRange="18400-24257" />
    <SegmentRl indexRange="162009-192028" mediaRange="162009-24257" />
    <SegmentRl indexRange="192029-202379" mediaRange="192029-24257" />
    <SegmentRl indexRange="202380-208028" mediaRange="202380-24257" />
    <SegmentRl indexRange="208029-70897" mediaRange="208029-24257" />
    <SegmentRl indexRange="70898-222529" mediaRange="70898-24257" />
  </SegmentList>
</Representation>

```

Figura 10: Video 320\*200

```
</Representation>
<Representation id="mp4" mimeType="video/mp4" codecs="avc3.640001" width="640" height="400" frameRate="30000/1001" sar="1:1" startWithSAP="0" bandwidth="409168">
  <SegmentList timescale="30000" duration="15000">
    <SegmentRl mediaRange="#922-22074" indexRange="922-957-22118"/>
    <SegmentRl mediaRange="#947-46923" indexRange="947-987-22118"/>
    <SegmentRl mediaRange="#947-46923" indexRange="9427-9472-39470"/>
    <SegmentRl mediaRange="#947-46923" indexRange="9427-9472-39470"/>
    <SegmentRl mediaRange="#78415-78565" indexRange="78415-78565"/>
    <SegmentRl mediaRange="#79339-80255" indexRange="79339-79382"/>
    <SegmentRl mediaRange="#80255-80309" indexRange="80255-80309"/>
    <SegmentRl mediaRange="#116469-11540" indexRange="#116469-116992"/>
    <SegmentRl mediaRange="#11340-11340" indexRange="#11340-11340"/>
    <SegmentRl mediaRange="#11340-146629" indexRange="#11340-113927"/>
    <SegmentRl mediaRange="#11340-146629" indexRange="#11340-113927"/>
    <SegmentRl mediaRange="#147598-156640" indexRange="#147598-147641"/>
    <SegmentRl mediaRange="#156658-225565" indexRange="#156658-156697"/>
    <SegmentRl mediaRange="#156658-225565" indexRange="#156658-156697"/>
    <SegmentRl mediaRange="#277721-327351" indexRange="#277721-277764"/>
    <SegmentRl mediaRange="#398236-398279" indexRange="#398236-398279"/>
    <SegmentRl mediaRange="#456858-506490" indexRange="#456858-456801"/>
    <SegmentRl mediaRange="#521196-557430" indexRange="#521196-557430"/>
    <SegmentRl mediaRange="#557431-558037" indexRange="#557431-557474"/>
  </SegmentList>
</Representation>
</VideoContent>
</Period>

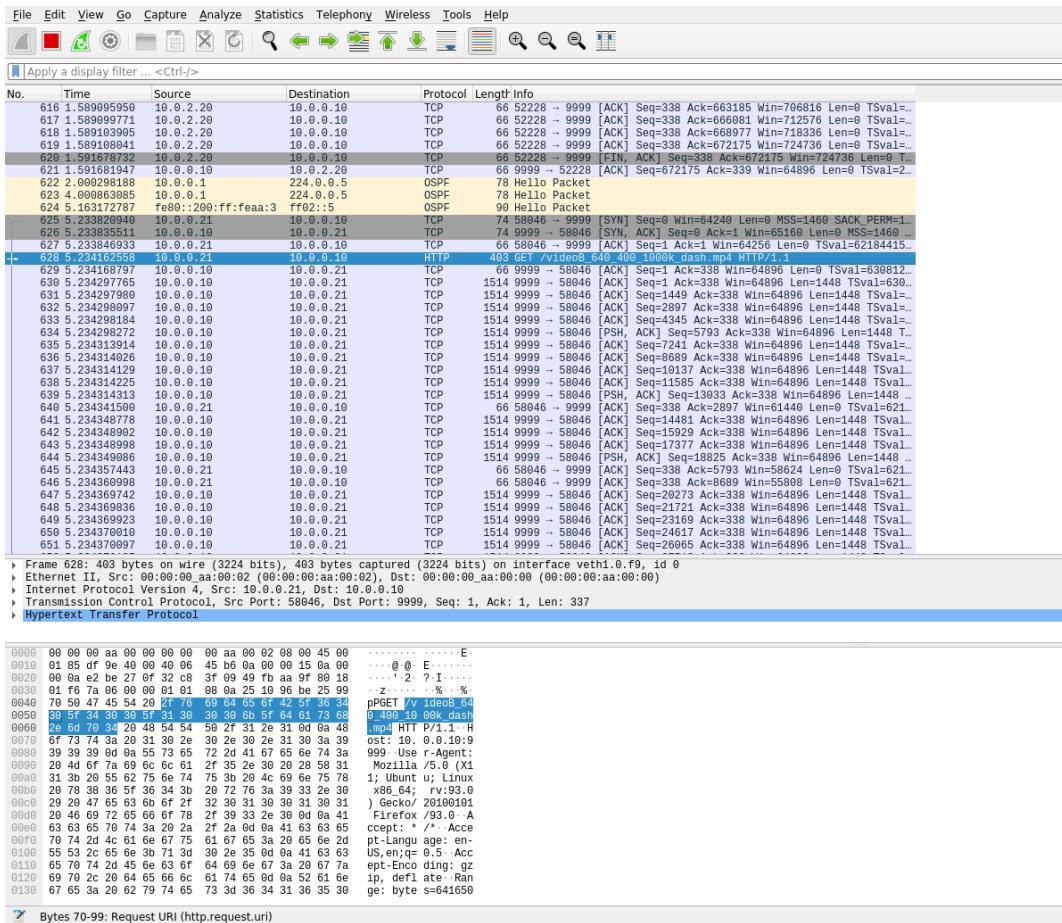
```

Figura 11: Video 640\*400

Neste cenário os protocolos usados, como ilustrado na figura 12, são :

- Protocolo IP (camada de rede).
- Protocolo Ethernet (camada de ligação de dados).
- Protocolo TCP ( camada de transporte).
- Protocolo HTTP (camada de aplicação).

Todos os protocolos que foram mencionados estão associados à rede global da *Internet*, uma vez que o vídeo foi transmitido a partir do *firefox*.



### 3 Questão 3

- 3.1 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

Para o portátil *Alladin* possa reproduzir o vídeo com maior resolução é necessário restringir o *link* na topologia, que já foi usada nas questões anteriores, de forma a permitir a transmissão do vídeo com a resolução 640x400. Para conseguir verificar a largura de banda necessária para a transmissão pode-se simplesmente realizar o comando: **cat video\_manifest.mpd**

Pode-se verificar que a largura de banda necessária para transmitir o vídeo é 409160 *bps* pelo que podemos inserir a restrição de largura de banda igual a 400000 *bps* no link do portátil *Alladin* de forma a não condicionar o resto da topologia com a restrição a ser inserida.

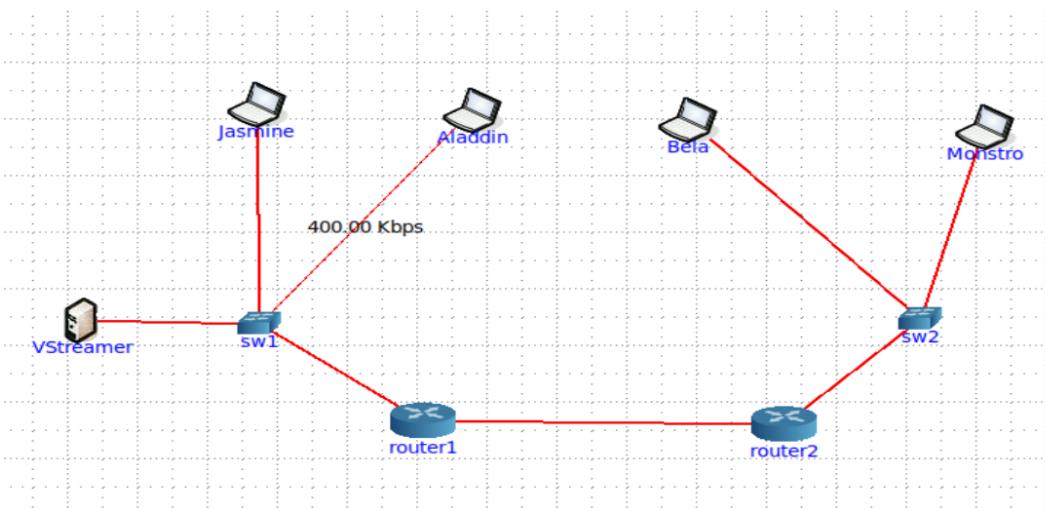


Figura 13: Topologia com a restrição para o portátil Alladin

Na captura do *wireshark* para o portátil *Alladin* (Figura 17) pode-se observar na linha 93 que foi transmitido o vídeo com maior resolução. Porém, devido ao funcionamento do *DASH* o vídeo vai flutuando entre resoluções.

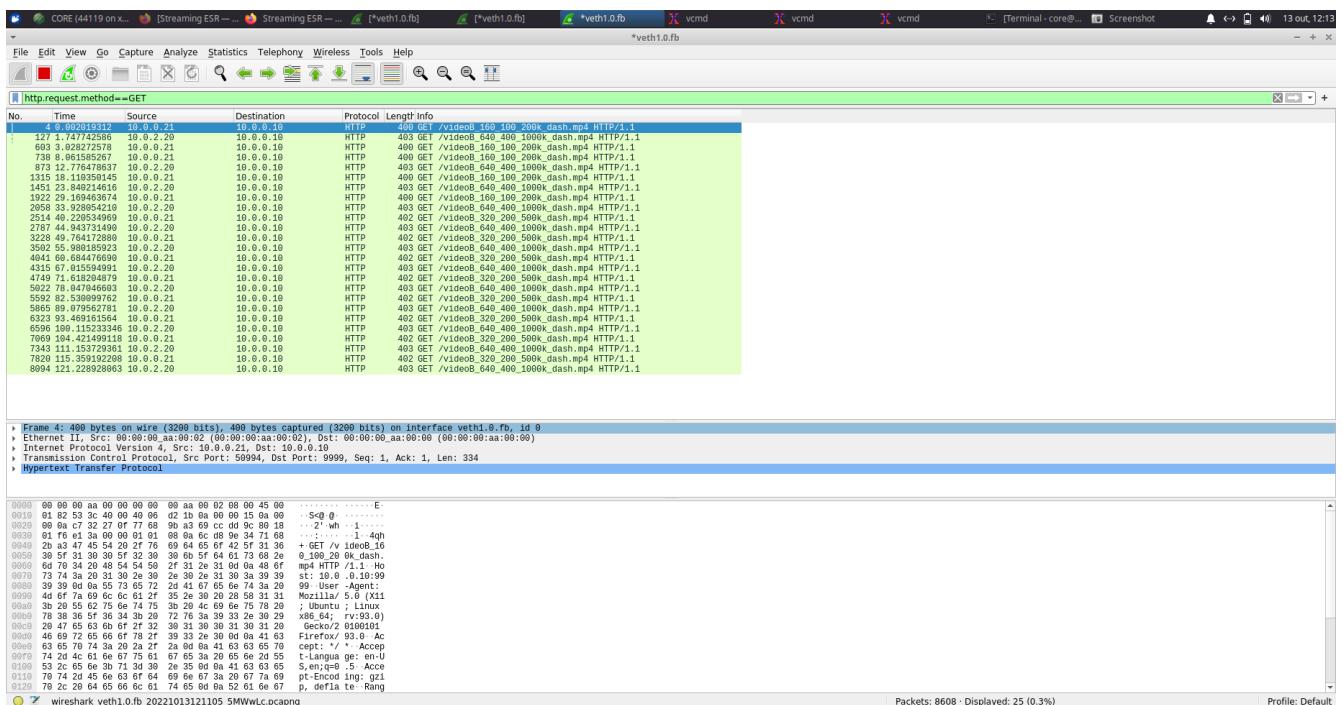


Figura 14: Wireshark do portátil *Alladin*

Para que portátil *Bela* possa reproduzir o video com menor resolução é necessário restringir o link na topologia. Mais uma vez verificando a informação contida no ficheiro video\_manifest.mpd pode-se observar que é necessário colocar uma restrição de 160000 *bps* no link do portátil *Bela*, visto que o segundo maior video tem uma largura de banda de 163532 *bps* e pretende-se não deixar passar mais nenhum video sem ser o video com resolução 160x100 que possui uma largura de banda inferior à restrição colocada.

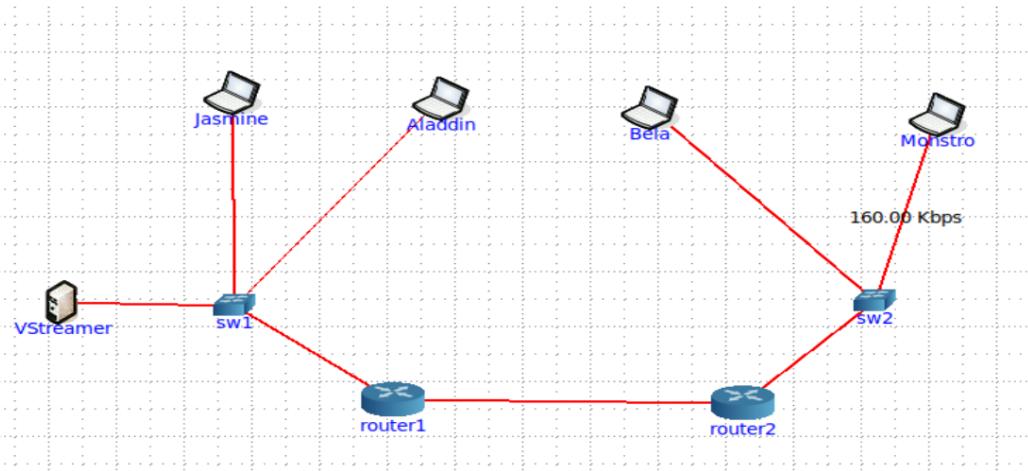


Figura 15: Topologia com a restrição para o portátil *Bela*

Portanto, na captura do wireshark do portátil *Bela* pode-se observar na linha 20920 a transmissão do video com menor resolução, tal como foi indicado anteriormente.

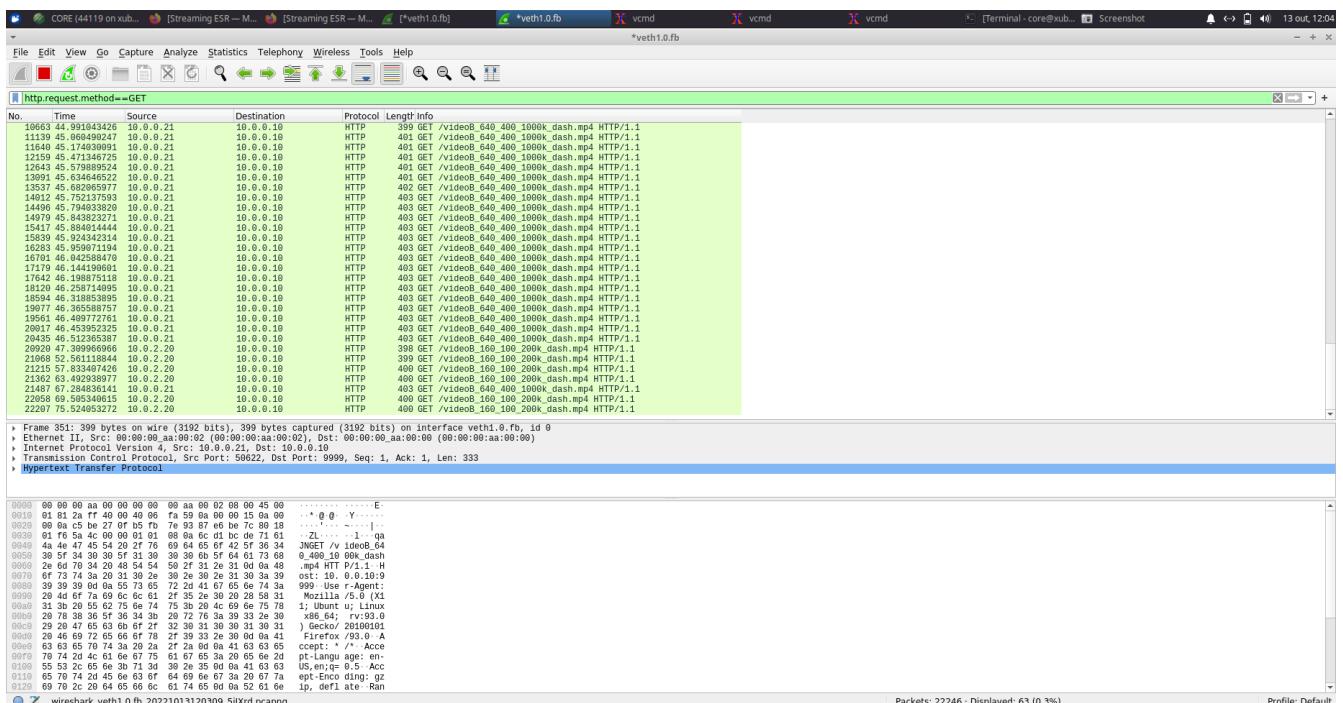


Figura 16: Wireshark do portátil *Bela*

## 4 Questão 4

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

O *Dynamic Adaptive Streaming Over HTTP*, ou *DASH*, é uma tecnologia de *streaming* com recurso a *HTTP* que permite a transmissão dinâmica e adaptativa às condições da rede. O *DASH* verifica a largura de banda do *link* e avalia qual é a melhor resolução de vídeo para ser transmitido nesse *link* recorrendo ao ficheiro "video\_manifest.mpd". Este ficheiro contém toda a informação sobre cada resolução disponível tal como a largura de banda necessária para transmitir o vídeo a cada resolução específica. Assim sendo, durante o *streaming* do vídeo, o *DASH* vai observando a qualidade da conexão e vai alterando dinamicamente a resolução do vídeo da forma mais eficiente possível.

Para exemplificar o funcionamento do *DASH*, pode-se observar quando foi realizado o *streaming* do vídeo no *firefox* no portátil *Alladin* e no portátil *Bela*. Durante o processo de *streaming* foram realizadas algumas capturas com o recurso ao *wireshark* nas quais é possível observar o *DASH* em acção.

Na Figura ?? pode-se verificar uma flutuação de resoluções como por exemplo nas linhas 603 e 738 que está a transmitir o vídeo com a resolução de 160x100 e depois na linha 873 já é transmitido o vídeo com resolução mais alta (640x400). Isto acontece pois o *DASH* verificou que a conectividade e/ou a transmissão na rede não eram indicadas para transmitir o vídeo com resolução mais elevada e por isso decidiu transmitir um vídeo com resolução mais baixa de forma a que perdas de *frames* ou pausas não seja notadas pelo utilizador que está a visualizar o vídeo.

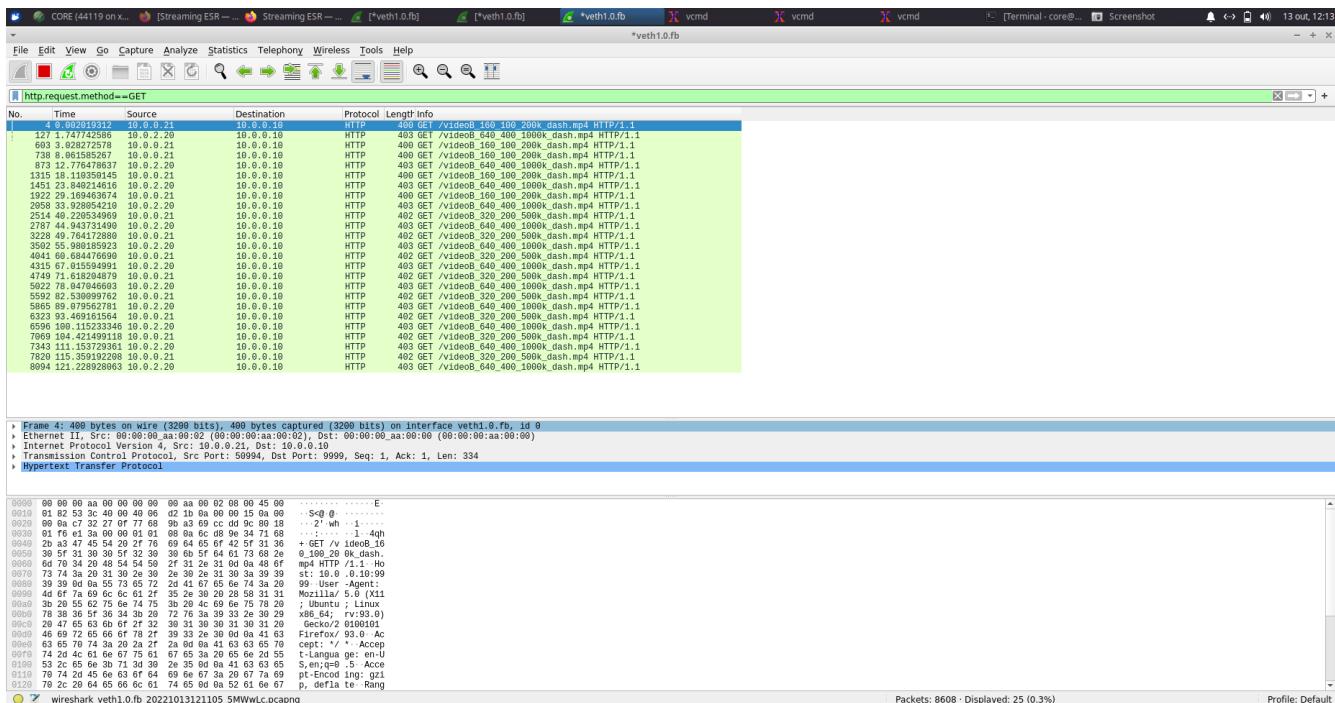


Figura 17: Wireshark do portátil *Alladin* demonstrando o *DASH*

## 5 Questão 5

**5.1 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.**

Primeiramente, esta etapa foi realizada numa máquina diferente à usada nas questões anteriores pelo que o ficheiro video utilizado é diferente dos anteriores. Para este efeito a imagem apresentada a seguir indica as informações sobre o *videoA.mp4* que foi usado nesta questão.

```
core@xubuncore:~$ 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-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 --arch=amd64 --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-libcodecs2 --enable-libdtsop --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libm3u8 --enable-libmyippy --enable-libopenjpeg --enable-libopenpnmpt --enable-libopus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-opengl --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
libavutil      56. 31.100 / 56. 31.100
libavcodec     58. 54.100 / 58. 54.100
libavformat    58. 29.100 / 58. 29.100
libavdevice    58.  8.100 / 58.  8.100
libavfilter     7. 57.100 / 7. 57.100
libavresample   4.  0. 0 / 4.  0. 0
libswscale      5.  5.100 / 5.  5.100
libswresample   3.  5.100 / 3.  5.100
libpostproc    55.  5.100 / 55.  5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoA.mp4':
  Metadata:
    major_brand : isom
    minor_version : 512
    compatible_brands: isomiso2avc1mp41
    encoder : Lavf58.29.100
  Duration: 00:00:11.50, start: 0.000000, bitrate: 25 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 160x100, 23 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
    Metadata:
      handler_name : VideoHandler
At least one output file must be specified
core@xubuncore:~$
```

Figura 18: *ffmpeg -i videoA.mp4*

Tanto num cenário *unicast* como *multicast* permitem a utilização do protocolo *RTP/RTCP*. Contudo, cada um destes cenários possui as suas vantagens e desvantagens.

No que toca ao ***unicast***, este consiste numa conexão "um para um", ou seja, este é composto por apenas um servidor e um cliente, o que leva a uma implementação mais "simples". Este tipo de conexão também corresponde ao tipo de tráfego mais comum que é o *TCP/IP*. Porém, num cenário *unicast* só é possível enviar *data* para um único destino.

O cenário ***multicast*** consiste numa conexão "um para vários", o que significa, que é composto por um servidor e vários clientes. *Multicast* é uma tecnologia capaz de enviar *data* da origem para vários pontos de destino distintos, geralmente estes pontos encontram-se dentro de redes locais. Neste cenário o tráfego da rede não aumenta, visto que os pacotes são enviados simultâneamente para vários clientes utilizando uma cópia do pacote a ser enviado, o que reduz significativamente o peso de envio. Relativamente à **escalabilidade**, pode-se afirmar que o cenário *multicast* é mais escalável que o *unicast* visto que a resposta do *streaming unicast* é mais eficiente quando existem vários clientes na rede. A maior desvantagem do *streaming unicast* em *multicast* é a elevada complexidade o que irá provocar maiores custos de controlo e um maior *overhead*.

Por fim, basta apenas observar que a quantidade de dados enviada em cada um dos cenários. Observando as figuras 19 e 20 pode-se verificar que cada transmissão é muito semelhante, 203k (*unicast*) e 205k (*multicast*). Contudo, o cenário *unicast* é relativo a um único cliente, ao contrário

do *multicast* que possui 3 clientes. Pode-se então concluir que o cenário *multicast* é muito mais escalável.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Frame	100.0	17539	100.0	15603151	213 k	0	0
Ethernet	100.0	17539	1.6	245546	3363	0	0
Internet Protocol Version 6	0.3	60	0.0	2400	32	0	0
Internet Control Message Protocol v6	0.0	2	0.0	32	0	2	32
Data	0.3	58	0.0	2088	28	58	2088
Internet Protocol Version 4	99.4	17435	2.2	348700	4776	0	0
User Datagram Protocol	97.6	17124	0.9	136992	1876	0	0
Data	97.2	17040	95.1	14837627	203 k	17040	14837627
ADwin configuration protocol	0.5	84	0.1	8354	114	84	8354
Internet Control Message Protocol	0.1	19	0.0	7332	100	19	7332
Data	1.7	292	0.1	12848	176	292	12848
Address Resolution Protocol	0.3	44	0.0	1232	16	44	1232

Figura 19: Cenário Unicast

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Frame	100.0	2636	100.0	2407440	216 k	0	0
Ethernet	100.0	2636	1.5	36904	3323	0	0
Internet Protocol Version 6	0.0	1	0.0	40	3	0	0
Internet Control Message Protocol v6	0.0	1	0.0	16	1	1	16
Internet Protocol Version 4	100.0	2635	2.2	52700	4745	0	0
User Datagram Protocol	100.0	2635	0.9	21080	1898	0	0
Session Announcement Protocol	0.7	18	0.2	5832	525	0	0
Session Description Protocol	0.7	18	0.2	5400	486	18	5400
Real-Time Transport Protocol	97.8	2579	94.8	2283437	205 k	0	0
MP4V-ES	97.8	2579	93.6	2252489	202 k	2579	2252489
Real-time Transport Control Protocol	0.7	18	0.0	504	45	18	504
Data	0.8	20	0.3	6927	623	20	6927

Figura 20: Cenário Multicast

## 6 Conclusão

Concluindo, com o desenvolvimento deste trabalho foi possível explorar várias técnicas e vários protocolos associados ao *streaming* de ficheiros vídeo e de audio, sendo que este guião apenas insidiu sobre ficheiros de vídeo. Foi possível analisar os tráfego e a escalabilidade de vários cenários de *streaming* de um vídeo em *HTTP* com várias resoluções diferentes e verificar e estudar o funcionamento do *DASH* que está encarregue de observar a conectividade e mudar a resolução do video, a ser transmitido, dinamicamente. Finalmente também foi feito um estudo sobre *unicast* e *multicast* pelo que nos permitiu aprender mais sobre o protocolo *UDP* com recurso a *RTP/RTCP*.