



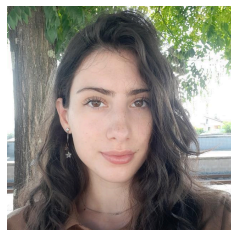
Universidade do Minho

Mestrado em Engenharia Informática

Unidade Curricular de Engenharia de Serviços em Rede

Ano Lectivo de 2024/2025

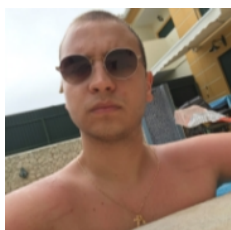
Grupo 85



Mariana Antunes Silva (PG55980)



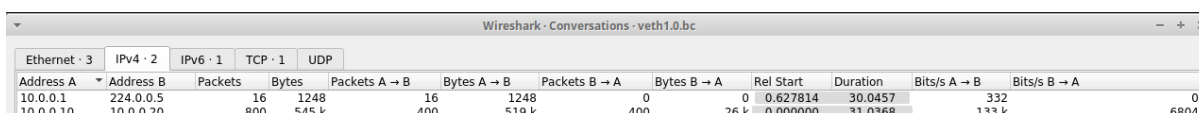
João Paulo Campelo Gomes (PG55960)



João Miguel Pinheiro Machado (PG53926)

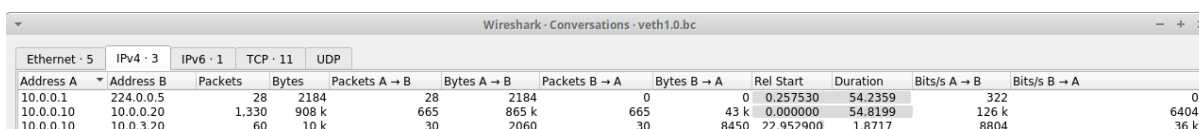
Questão 1

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).



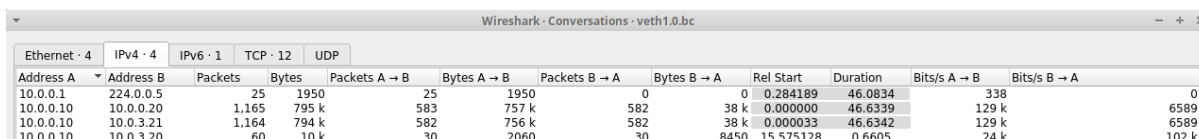
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.1	224.0.0.5	16	1248	16	1248	0	0	0.627814	30.0457	332	0
10.0.0.10	10.0.0.20	800	545 k	400	519 k	400	26 k	0.000000	31.0368	133 k	6804

1.1.Amostra de tráfego no link de saída do servidor -> 1 clientes (VLC)



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	28	2184	28	2184	0	0	0.257530	54.2359	322	0
10.0.0.10	10.0.0.20	1,330	908 k	665	865 k	665	43 k	0.000000	54.8199	126 k	6404
10.0.0.10	10.0.3.20	60	10 k	30	2060	30	8450	22.952900	1.8717	8804	36 k

1.2.Amostra de tráfego no link de saída do servidor -> 2 clientes (VLC e Firefox)



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	25	1950	25	1950	0	0	0.284189	46.0834	338	0
10.0.0.10	10.0.0.20	1,165	795 k	583	757 k	582	38 k	0.000000	46.6339	129 k	6589
10.0.0.10	10.0.3.21	1,164	794 k	582	756 k	582	38 k	0.000033	46.6342	129 k	6589
10.0.0.10	10.0.3.20	60	10 k	30	2060	30	8450	15.575128	0.6605	24 k	102 k

1.3.Amostra de tráfego no link de saída do servidor -> 3 clientes (VLC, Firefox, ffmpeg)

a) Comente os protocolos utilizados na transferência, bem como a experiência que o utilizador terá caso o link utilizado tenha perdas.

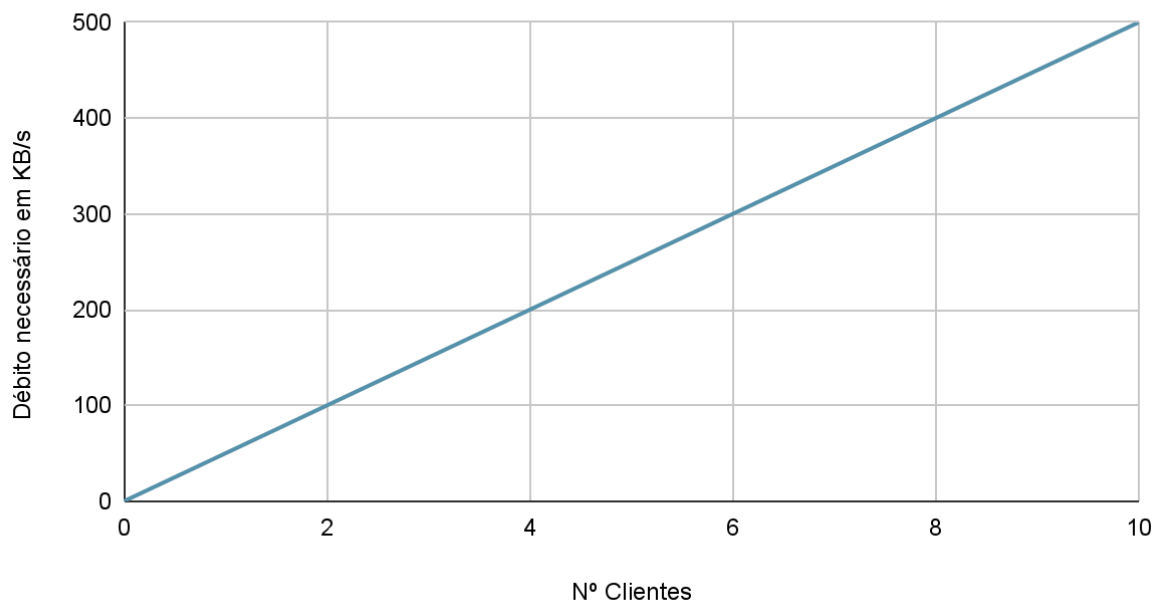
O protocolo de transferência utilizado é o TCP, que garante a entrega dos pacotes através do reenvio em caso de falha. No entanto, este protocolo tem um overhead elevado, o que significa que é necessário transferir um maior número de bytes em comparação com o UDP. Se ocorrerem falhas na ligação, o protocolo TCP faz com que o servidor de streaming interrompa o envio de novos pacotes até que o pacote em falta seja recebido com sucesso. Isto faz com que a transmissão no cliente fique congelada no último frame enviado e retome assim que o pacote em falta for entregue. Como consequência, o cliente passará a ver a transmissão com um atraso, que aumentará com cada novo pacote perdido.

b) Identifique o número total de fluxos gerados e elabore um gráfico que demonstre a evolução do débito dependendo do número de clientes.

O número de fluxos gerados é igual ao número de clientes que estão a ver a stream, neste caso como há três clientes há três fluxos de transferência.

Assumindo um débito mínimo necessário de 50kb/s para um cliente, o gráfico de evolução do débito seria algo semelhante a este:

Evolução do débito



1.4. Evolução do débito

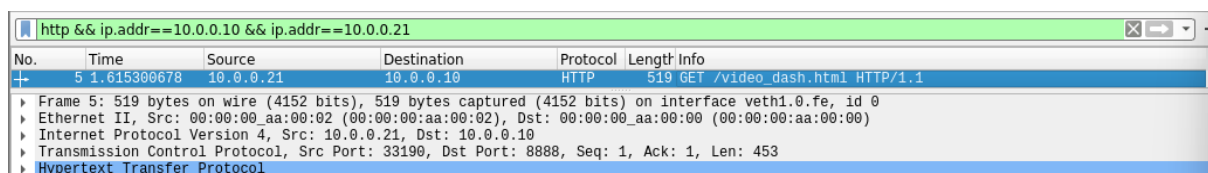
- c) **Comente a escalabilidade da solução para 1000 utilizadores, assim como 10000 utilizadores. Crie uma expressão matemática que expresse o débito necessário para que o servidor envie vídeo para N clientes.**

Como para cada cliente é necessária a criação de um fluxo novo, a cada cliente adicionado o débito necessário aumenta de forma linear. O débito total necessário para n clientes, pode ser obtido através da seguinte fórmula: $\text{débito total} = \text{débito para um cliente} * n$. Tendo isto em conta esta arquitetura não é muito escalável, uma vez que se assumirmos um débito de 100KB/s necessários para um cliente, para 1000 é necessário um débito total de 100 MB/s o que causaria imenso stress na rede, mas talvez ainda seria possível. No entanto, com 10000 utilizadores seria necessário um débito de 1 GB/s o que é completamente inviável.

Questão 2

Utilize o wireshark para determinar 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. Explique como obteve esta informação.

A pilha protocolar usada é a HTTP/Ethernet/IP/TCP, como podemos ver no print tirado no Wireshark.



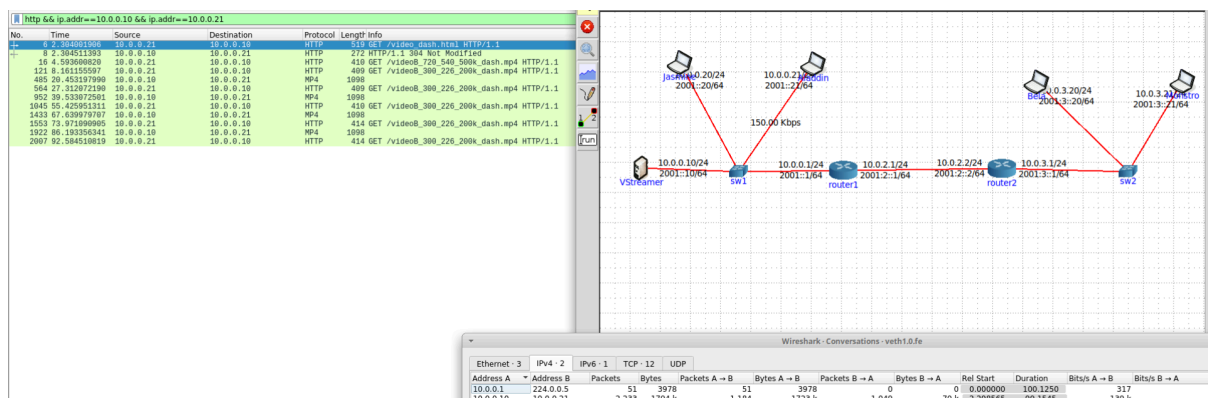
The image shows a Wireshark packet capture window. The filter bar at the top is set to 'http && ip.addr==10.0.0.10 && ip.addr==10.0.0.21'. The packet list shows a single packet (No. 5) at time 1.615300678, from source 10.0.0.21 to destination 10.0.0.10, protocol HTTP, length 519. The packet details pane shows the following layers: Ethernet II (Src: 00:00:00:aa:00:02, Dst: 00:00:00:aa:00:00), Internet Protocol Version 4 (Src: 10.0.0.21, Dst: 10.0.0.10), Transmission Control Protocol (Src Port: 33190, Dst Port: 8888, Seq: 1, Ack: 1, Len: 453), and Hypertext Transfer Protocol (GET /video_dash.html HTTP/1.1).

No.	Time	Source	Destination	Protocol	Length	Info
5	1.615300678	10.0.0.21	10.0.0.10	HTTP	519	GET /video_dash.html HTTP/1.1

2.1. Pilha protocolar

- **Vídeo 300 x 226**

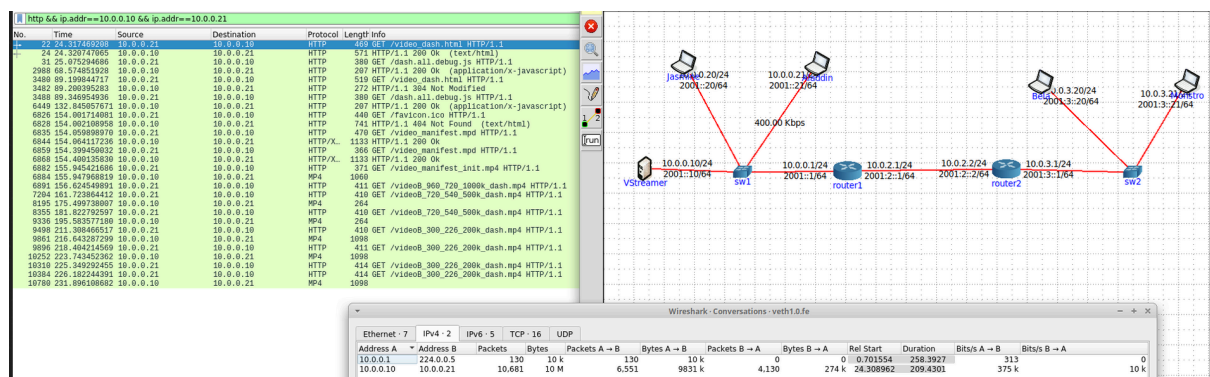
Para podermos visualizar a largura de banda de um vídeo com a resolução com 300 x 226 ajustamos o débito dos links de topologia para 150Kbps de modo que o cliente no portátil Aladdin exiba o vídeo com a resolução pretendida, depois de limitar a largura de banda verificada no conversations do wireshark tem valor de 139 Kbps.



2.2. Conversations do vídeo 300 x 226

- **Vídeo 720 x 540**

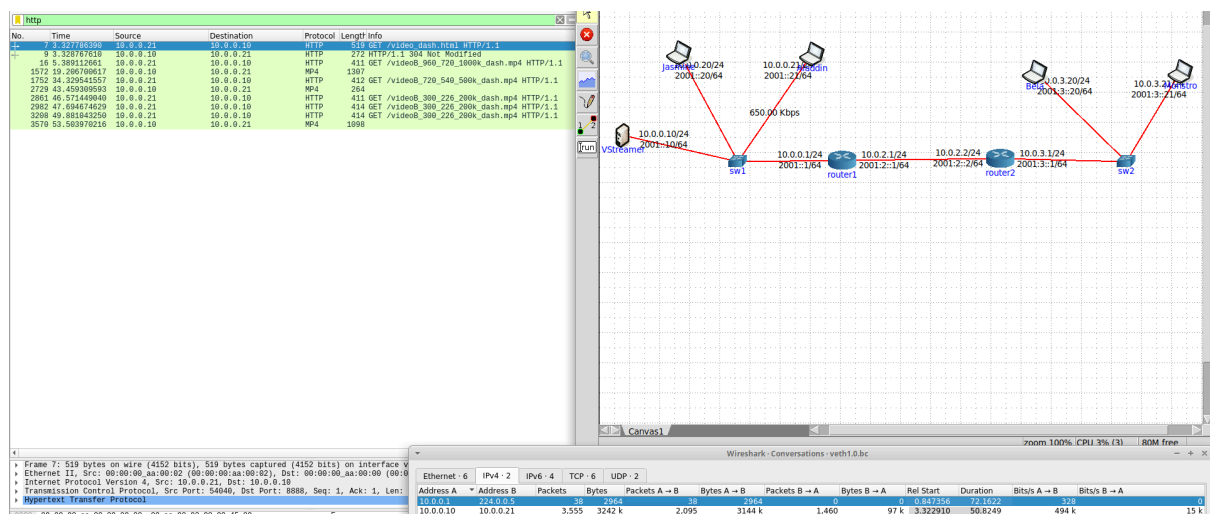
Da mesma forma descrita acima ajustamos o débito dos links da topologia para 400Kbps e concluímos que a largura de banda verificada tem valor de 375 Kbps.



2.3. Conversations do vídeo 720 x 540

• Vídeo 960 x 720

Para podermos visualizar a largura de banda necessária para um vídeo de resolução 960 x 720 é necessário limitar o débito do link a 650 Kbps e verificamos que a largura de banda necessária é igual a 494 Kbps. O valor real é mais alto do que o medido, pois o vídeo enviado só tem 21 segundos, mas fizemos a medição ao fim de 50 segundos o que leva a que este valor tenha baixado, devido ao tempo que houve sem que se estivesse a enviar dados do mesmo.



2.4. Conversations do vídeo 960 x 720

Questão 3

Compare a largura de banda medida na questão anterior com a que é disponibilizada pelo ffmpeg. Qual é a razão para a diferença entre as duas?

Tendo em conta a largura de banda obtida pelo Wireshark mostrada na pergunta anterior, concluímos que para todas as resoluções, a largura de banda medida pelo wireshark é maior que a bitrate dada pelo ffmpeg, pois o ffmpeg apenas

analisa o ficheiro, utilizando os frames per second do vídeo e a sua resolução calcula a taxa de transferência necessária para visualizar esses frames por segundo. Como estamos a fazer stream deste vídeo pela internet, para além dos bits do ficheiro em si, também temos de enviar os cabeçalhos que os protocolos necessitam, o que leva à discrepância mencionada anteriormente. O valor obtido na alínea anterior para a maior resolução é menor que a bitrate, mas isso deve-se apenas a um erro de medição, pois o vídeo enviado tinha 21s, mas fizemos a medição ao fim de 50 segundos o que resultou com que o valor medido fosse menor devido ao tempo onde não foram enviados dados do vídeo.

- **Vídeo 300 x 226 -> 124 kb/s**

```
core@xubuncore:~$ ffplay -stats videoB_300_226_200k.mp4
ffplay version 4.2.7-0ubuntu0.1 Copyright (c) 2003-2022 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 --incdir=/usr/include/x86_64-linux-gnu --arch=
  amd64 --enable-gpl --disable-stripping --enable-avresample --disable-filter=resa
  mple --enable-avisynth --enable-gnutls --enable-ladspa --enable-libaom --enable-
  libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --e
  nable-libcodec2 --enable-libflite --enable-libfontconfig --enable-libfreetype --
  enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libm
  p3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libo
  pus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine
  --enable-libsnapppy --enable-libsoxr --enable-libspeex --enable-libssh --enable-l
  ibtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-lib
  vpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --ena
  ble-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-
  openal --enable-openccl --enable-opengl --enable-sdl2 --enable-libdc1394 --enable
  libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --enable-frei0r
  --enable-libx264 --enable-shared
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter    7. 57.100 / 7. 57.100
  libavresample  4.  0.  0 / 4.  0.  0
  libswscale     5.  5.100 / 5.  5.100
  libswresample  3.  5.100 / 3.  5.100
  libpostproc   55.  5.100 / 55.  5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoB_300_226_200k.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf58.29.100
  Duration: 00:00:20.93, start: 0.000000, bitrate: 124 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 300x226,
  121 kb/s, 29.92 fps, 29.92 tbr, 11488 tbn, 59.83 tbc (default)
  Metadata:
    handler_name     : VideoHandler
  2.71 M-V: -0.033 fd=  4 aq=   0KB vq=  20KB sq=   0B f=0/0
```

3.1. Comando ffplay do vídeo 300 x 226

- **Video 720 x 540 -> 354 kb/s**

```
core@xubuncore:~$ ffplay -stats videoB_720_540_500k.mp4
ffplay version 4.2.7-0ubuntu0.1 Copyright (c) 2003-2022 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 --incdir=/usr/include/x86_64-linux-gnu --arch=
  amd64 --enable-gpl --disable-stripping --enable-avresample --disable-filter=res
  ample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libaom --enable-
  libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --e
  nable-libcodec2 --enable-libflite --enable-libfontconfig --enable-libfreetype --
  enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libm
  p3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libo
  pus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine
  --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-l
  ibtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-lib
  vpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --ena
  ble-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-
  opengl --enable-openc1 --enable-opengl --enable-sdl2 --enable-libdc1394 --enable
  -libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --enable-frei0r
  --enable-libx264 --enable-shared
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale     5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoB_720_540_500k.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf58.29.100
  Duration: 00:00:20.93, start: 0.000000, bitrate: 354 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 720x540,
  352 kb/s, 29.92 fps, 29.92 tbr, 11488 tbn, 59.83 tbc (default)
  Metadata:
    handler_name     : VideoHandler
  2.46 M-V: -0.033 fd=  6 aq=   0KB vq=  51KB sq=   0B f=0/0
```

3.2. Comando ffplay do vídeo 720 x 540

- **Video 960 x 720 -> 565 kb/s**

```

core@xubuncore:~$ ffplay -stats videoB_960_720_1000k.mp4
ffplay version 4.2.7-0ubuntu0.1 Copyright (c) 2003-2022 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 --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-libcodec2 --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-libsvg --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-lsbspeex --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-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
 libavutil      56. 31.100 / 56. 31.100
 libavcodec     58. 54.100 / 58. 54.100
 libavformat    58. 29.100 / 58. 29.100
 libavdevice    58.  8.100 / 58.  8.100
 libavfilter    7. 57.100 / 7. 57.100
 libavresample  4.  0.  0 / 4.  0.  0
 libswscale     5.  5.100 / 5.  5.100
 libswresample  3.  5.100 / 3.  5.100
 libpostproc   55.  5.100 / 55.  5.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoB_960_720_1000k.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf58.29.100
  Duration: 00:00:20.93, start: 0.000000, bitrate: 565 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 960x720, 562 kb/s, 29.92 fps, 29.92 tbr, 11488 tbn, 59.83 tbc (default)
  Metadata:
    handler_name     : VideoHandler
 2.48 M-V:  0.002 fd= 31 aq=  0KB vq=  83KB sq=  0B f=0/0

```

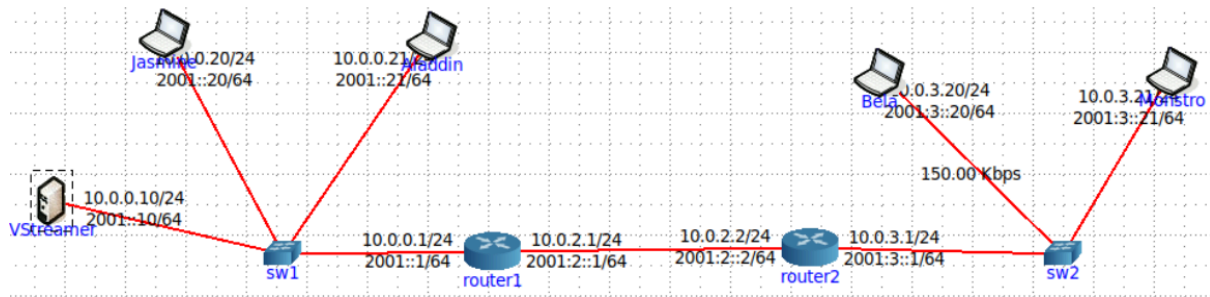
3.3. Comando ffplay do vídeo 960 x 720

Questão 4

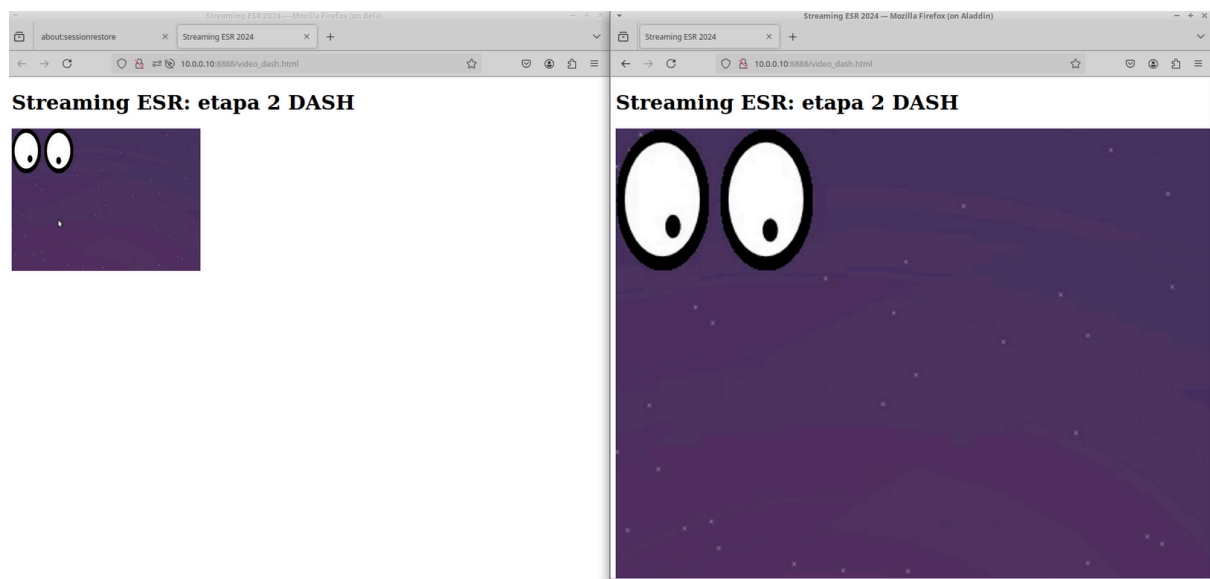
Ajuste o débito dos links da topologia de modo que o cliente no portátil Bela exiba o vídeo de menor resolução e o cliente no portátil Alladin exiba o vídeo com mais resolução. Mostre evidências e justifique a largura de banda necessária para que o stream de vídeo sofra alterações.

Para exibir o vídeo de menor resolução no portátil da Bela foi necessário limitar a largura de banda a 150 Kbps no link que conecta a Bela, pois como verificamos a largura de banda disponibilizada pelo ffplay para exibir o vídeo de resolução 300 x 226 é necessário ter uma largura de banda [124, 354] kb/s.

Para o Aladdin exibir o vídeo de maior resolução 960 x 720 é necessário de ter uma largura de banda segundo o ffplay superior a 565 kb/s, para tal não foi necessário limitar o débito dos links da topologia para ter o efeito pretendido.



4.1. Topologia com BandWidth no link que conecta a Bela a 150 Kbps



4.2. Resolução da *stream* (Bela vs Alladin)

No.	Time	Source	Destination	Protocol	Length	Info
10	8.554896048	10.0.3.20	10.0.0.10	HTTP	519	GET /video_dash.html HTTP/1.1
12	8.555230273	10.0.0.10	10.0.3.20	HTTP	272	HTTP/1.1 304 Not Modified
18	9.880876558	10.0.3.20	10.0.0.10	HTTP	410	GET /videoB_720_540_500k_dash.mp4 HTTP/1.1
7845	15.559841718	10.0.3.20	10.0.0.10	HTTP	409	GET /videoB_300_226_200k_dash.mp4 HTTP/1.1
33801	27.835453529	10.0.0.10	10.0.3.20	MP4	1098	
45292	34.594488393	10.0.3.20	10.0.0.10	HTTP	409	GET /videoB_300_226_200k_dash.mp4 HTTP/1.1
47054	46.807857235	10.0.0.10	10.0.3.20	MP4	1098	
47146	62.824754033	10.0.3.20	10.0.0.10	HTTP	410	GET /videoB_300_226_200k_dash.mp4 HTTP/1.1
47535	75.043303949	10.0.0.10	10.0.3.20	MP4	1098	
47655	81.368343436	10.0.3.20	10.0.0.10	HTTP	414	GET /videoB_300_226_200k_dash.mp4 HTTP/1.1
48024	93.596969951	10.0.0.10	10.0.3.20	MP4	1098	
48119	99.986733571	10.0.3.20	10.0.0.10	HTTP	414	GET /videoB_300_226_200k_dash.mp4 HTTP/1.1
48489	112.312775502	10.0.0.10	10.0.3.20	MP4	1098	

4.3. Pedido http entre Bela e VStreamer

http && ip.addr==10.0.0.10 && ip.addr==10.0.0.21					
No.	Time	Source	Destination	Protocol	Length Info
61	10.675198137	10.0.0.21	10.0.0.10	HTTP	519 GET /video_dash.html HTTP/1.1
63	10.675548382	10.0.0.10	10.0.0.21	HTTP	272 HTTP/1.1 304 Not Modified
102	11.822621289	10.0.0.21	10.0.0.10	HTTP	411 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
1183	11.883525853	10.0.0.10	10.0.0.21	MP4	1307
1203	12.408125677	10.0.0.21	10.0.0.10	HTTP	413 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
2288	12.478894894	10.0.0.10	10.0.0.21	MP4	1307
2306	12.935593942	10.0.0.21	10.0.0.10	HTTP	413 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
3387	13.098633094	10.0.0.10	10.0.0.21	MP4	1307
3408	13.576990485	10.0.0.21	10.0.0.10	HTTP	414 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
4493	13.688115494	10.0.0.10	10.0.0.21	MP4	1307
4510	14.202548576	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
5624	14.305062220	10.0.0.10	10.0.0.21	MP4	1307
5644	14.901940416	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
6734	14.987754246	10.0.0.10	10.0.0.21	MP4	1307
6745	15.350819226	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
7837	15.464484067	10.0.0.10	10.0.0.21	MP4	1307
7868	15.765642687	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
8952	15.899274191	10.0.0.10	10.0.0.21	MP4	1307
8987	16.615780870	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
10076	16.700949668	10.0.0.10	10.0.0.21	MP4	1307
10100	17.400131891	10.0.0.21	10.0.0.10	HTTP	415 GET /videoB_960_720_1000k_dash.mp4 HTTP/1.1
11183	17.513195974	10.0.0.10	10.0.0.21	MP4	1307

4.4. Pedido http entre Aladdin e VStreamer

Questão 5

Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado e comparando o modelo de streaming com o que foi utilizado na Questão 1.

O protocolo DASH (*Dynamic Adaptive Streaming over HTTP*), é um protocolo de transmissão de vídeo que permite a entrega adaptativa de conteúdo multimédia pela internet. Uma das principais vantagens do DASH é ajustar a qualidade do vídeo de acordo com as condições da rede e da capacidade do dispositivo e garantir uma experiência de visualização contínua e sem interrupções.

O ficheiro MPD é um ficheiro XML que desempenha um papel central no funcionamento do DASH, pois este contém todas as informações necessárias para que o cliente possa localizar, conectar e reproduzir o conteúdo multimédia adaptativamente. O ficheiro contém também informações sobre as regras de adaptação, como a lógica para selecionar a qualidade apropriada com base nas condições de rede do cliente.

Questão 6

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 também para os cenários de 1000 e 10000 clientes.

Unicast e multicast são dois métodos de transmissão de dados em redes. No unicast, cada cliente recebe uma cópia individual dos dados diretamente do servidor, o que resulta em múltiplos fluxos de dados, aumentando o consumo de

largura de banda à medida que o número de utilizadores cresce. Já no multicast, o servidor envia uma única cópia dos dados para um grupo de clientes, e os pacotes são replicados pela rede apenas quando necessário, tornando-o muito mais eficiente em termos de largura de banda, especialmente para grandes audiências.

No Unicast cada cliente que se conecta ao servidor recebe uma cópia individual da stream, o que pode ser ineficiente em termos de largura de banda, especialmente à medida que o número de clientes aumenta. Cada cliente gera um fluxo independente de dados, aumentando significativamente o tráfego na rede conforme o número de clientes cresce. Para cenários com muitos utilizadores (1000 ou 10000), o modelo unicast não escala bem, pois a largura de banda exigida aumenta linearmente com o número de clientes, levando ao esgotamento de recursos de rede e do servidor.

Em contraste, o Multicast permite que o servidor envie uma única cópia da stream para um endereço de grupo multicast, e os routers da rede replicam os pacotes apenas quando necessário, de forma a alcançar todos os clientes que se inscreveram nesse grupo multicast. Isso reduz drasticamente o tráfego na rede, independentemente do número de clientes. A escalabilidade é uma das principais vantagens do multicast, uma vez que a largura de banda usada é constante, independentemente do número de utilizadores. Este modelo torna-se muito mais eficiente para grandes quantidades de clientes, como 1000 ou 10000, sem sobrecarregar a rede ou o servidor. Algumas vantagens do Multicast são: Escalabilidade, pois este lida muito melhor com um grande número de clientes, pois o servidor não precisa de enviar múltiplas cópias da stream; Eficiência, pois este gera menos tráfego de rede globalmente, uma vez que os pacotes são replicados apenas quando necessário. Algumas desvantagens são: Configuração da rede, a implementação do multicast pode ser mais complexa, pois requer suporte no roteamento da rede e nem todas as redes estão configuradas para suportar multicast; Controlo mais complexo, requerendo gerir e monitorizar os fluxos multicast pode ser mais difícil do que no unicast.

Em suma, o multicast é claramente superior em termos de escalabilidade e eficiência de largura de banda para grandes audiências. O unicast, embora mais simples, não é adequado para cenários com muitos utilizadores devido à sua ineficiência no uso da largura de banda.

Wireshark - Protocol Hierarchy Statistics - Etapa 3 Unicast.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	1532	100.0	1254091	185 k	0	0	0
▼ Ethernet	100.0	1532	1.7	21448	3174	0	0	0
▼ Internet Protocol Version 6	0.5	8	0.0	320	47	0	0	0
Open Shortest Path First	0.4	6	0.0	216	31	6	216	31
Internet Control Message Protocol v6	0.1	2	0.0	32	4	2	32	4
▼ Internet Protocol Version 4	99.2	1520	2.4	30400	4500	0	0	0
▼ User Datagram Protocol	97.5	1493	1.0	11944	1768	0	0	0
Real-time Transport Control Protocol	0.7	11	0.0	308	45	11	308	45
Data	93.9	1438	94.4	1183643	175 k	1438	1183643	175 k
ADwin configuration protocol	2.9	44	0.4	4480	663	44	4480	663
Open Shortest Path First	1.8	27	0.1	1188	175	27	1188	175
Address Resolution Protocol	0.3	4	0.0	112	16	4	112	16

No display filter.

Help Copy Close

6.1.Pacotes enviados em Unicast

Wireshark - Protocol Hierarchy Statistics - Etapa 3 Multicast.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	1413	100.0	1155452	182 k	0	0	0
▼ Ethernet	100.0	1413	1.7	19782	3118	0	0	0
▼ Internet Protocol Version 4	100.0	1413	2.4	28260	4454	0	0	0
▼ User Datagram Protocol	100.0	1413	1.0	11304	1781	0	0	0
Session Announcement Protocol	0.7	10	0.3	3240	510	0	0	0
Session Description Protocol	0.7	10	0.3	3000	472	10	3000	472
▼ Real-Time Transport Protocol	96.4	1362	93.2	1076464	169 k	0	0	0
MP4V-ES	96.4	1362	91.7	1060120	167 k	1362	1060120	167 k
Real-time Transport Control Protocol	0.7	10	0.0	280	44	10	280	44
Data	2.1	29	1.4	15914	2508	29	15914	2508
ADwin configuration protocol	0.1	2	0.0	208	32	2	208	32

No display filter.

Help Copy Close

6.2.Pacotes enviados em Multicast

No cenário de 1000 utilizadores ao unicast teria de realizar 1000 ligações diferentes cada uma delas enviado uma cópia do vídeo por cada ligação o que leva a um enorme aumento de banda larga, sobrecarregar o servidor e aumentar o tráfego da rede, porém por multicast , a situação seria muito mais eficiente do que no unicast, pois em vez de o servidor gerar 1000 fluxos de dados separados, ele enviaria apenas uma única stream para um grupo multicast, e a rede, por sua vez, seria responsável por replicar os pacotes apenas quando necessário, à medida que os utilizadores se conectam. No cenário de 10000 utilizadores, por unicast o impacto

seria ainda maior, levando a consumo massivo de largura de banda, sobrecarga extrema no servidor e o congestionamento da rede. De novo, por multicast, a situação seria muito mais eficiente em termos de uso de largura de banda e carga no servidor. A infraestrutura de rede replica os pacotes conforme necessário, tornando-o uma escolha muito mais escalável e sustentável para um elevado número de utilizadores.

Conclusões

Ao longo deste trabalho, exploramos diferentes abordagens de streaming de áudio e vídeo, compreendendo os impactos na rede e a eficiência das soluções estudadas. Na Etapa 1, o streaming via HTTP sem adaptação revelou as limitações do unicast, que não escala bem para grandes audiências, gerando sobrecarga no servidor e tráfego excessivo na rede. Já na Etapa 2, com o MPEG-DASH, vimos como a adaptação de débito otimiza a experiência do utilizador, ajustando a qualidade do vídeo conforme a largura de banda disponível. Essa técnica mostrou-se eficaz em fornecer um streaming contínuo e eficiente, mesmo com variações na rede.

Na Etapa 3, analisamos o RTP/RTCP em unicast e multicast, destacando a superioridade do multicast para grandes volumes de utilizadores, com uma eficiência de largura de banda significativamente maior. Em resumo, este trabalho proporcionou uma compreensão prática das diferentes soluções de streaming, consolidando a importância da escolha da pilha protocolar adequada para garantir escalabilidade, qualidade de serviço e eficiência de rede.