Engenharia de Serviços em Rede
Trabalho Prático 2: Nível Aplicacional: Conceitos Introdutórios

José Santos, Leonardo Marreiros, and Pedro Fernandes

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a84288, pg47398, pg47559}@alunos.uminho.pt

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 ffmeg). Identifique a taxa em bps necessária (usando o ffmpeg -i video1.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)

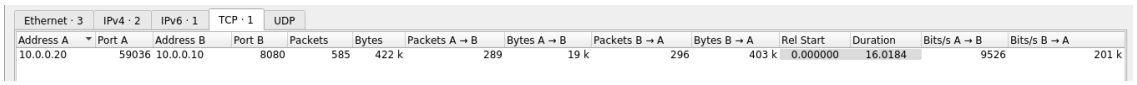
A *bitrate* do vídeo gerado é 44000bps [Figura 1], fazendo assim com que a taxa necessária seja maior ou igual à *bitrate* do vídeo. Nesta solução é usado o TCP como encapsulamento e é gerado um fluxo por cliente. Esta solução não é muito escalável tendo em conta que o ficheiro está apenas num servidor, com apenas uma qualidade, e portanto, no caso em que a *bitrate* do vídeo é maior do que a largura de banda, não há um *resize* do vídeo , ficamos apenas em *buffering* e com uma experiência de *streaming* pouco favorável. Sendo que estamos a falar de larga escala, não é de todo a solução mais indicada.

```
core@xubuncore:~$ ffmpeg -i video1.mp4
ffmpeg version 4.2.4-1ubuntu0.1 Copyright (c) 2000-2020 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.3.0-10ubuntu2)
  configuration: --prefix=/usr --extra-version=1ubuntu0.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/incl
ude/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-libgs
m --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpuls
e --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-libwavpack --enable-libwebp --e
nable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-
opencl --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-nvenc --enable-chromaprint --e
nable-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 'video1.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.29.100
  Duration: 00:00:13.15, start: 0.000000, bitrate: 44 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 180x120, 42 kb/s, 20 fps, 20 tbr, 10240 tbn, 40 tbc (default)
    Metadata:
      handler_name    : VideoHandler
```

Figura 1: Bitrate necessária para o *streaming* do video 1

Foi capturado tráfego com o *Wireshark* para cada uma das três situações. Utilizando estes dados, e fazendo uso da funcionalidade *Statistics* > *Conversations* do programa e filtrando por TCP, conseguimos

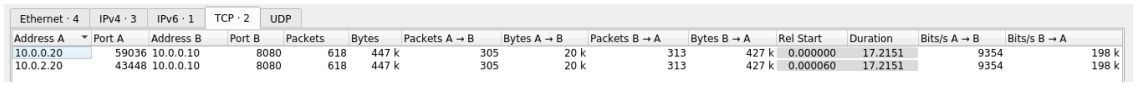
ter acesso ao número de fluxos e também à largura de banda em bps transmitida do servidor para os clientes. Obtemos os seguintes resultados:



Ethernet · 3 | IPv4 · 2 | IPv6 · 1 | TCP · 1 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	59036	10.0.0.10	8080	585	422 k	289	19 k	296	403 k	0.000000	16.0184	9526	201 k

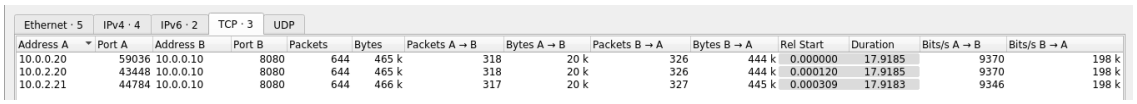
Figura 2: Dados referentes ao *streaming* do video 1 com um cliente



Ethernet · 4 | IPv4 · 3 | IPv6 · 1 | TCP · 2 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	59036	10.0.0.10	8080	618	447 k	305	20 k	313	427 k	0.000000	17.2151	9354	198 k
10.0.2.20	43448	10.0.0.10	8080	618	447 k	305	20 k	313	427 k	0.000060	17.2151	9354	198 k

Figura 3: Dados referentes ao *streaming* do video 1 com dois cliente



Ethernet · 5 | IPv4 · 4 | IPv6 · 2 | TCP · 3 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	59036	10.0.0.10	8080	644	465 k	318	20 k	326	444 k	0.000000	17.9185	9370	198 k
10.0.2.20	43448	10.0.0.10	8080	644	465 k	318	20 k	326	444 k	0.000120	17.9185	9370	198 k
10.0.2.21	44784	10.0.0.10	8080	644	466 k	317	20 k	327	445 k	0.000309	17.9183	9346	198 k

Figura 4: Dados referentes ao *streaming* do video 1 com três cliente

Com isto, podemos observar que a largura de banda distribuída foi sempre por volta de 200kbps, isto significa que houve um *overhead* de 200 - 44 = 156kbps, o que significa um débito muito maior daquele que seria necessário para fazer *streaming* do vídeo. Com isto podemos concluir que, com HTTP simples sem adaptação dinâmica do débito, uma vez que a largura de banda distribuída para cada cliente é igual, a escalabilidade deste método é linear, e acarreta um *overhead* desnecessário pelo que é um método muito pouco eficiente.

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.

Neste caso, como foram criada várias versões do vídeo 2, a largura de banda necessária para que o cliente consiga receber o vídeo terá de ser igual ou maior que o *bitrate* do vídeo de menor resolução, ou seja, pelo menos 162kbps. [Figura 5]

A pilha protocolar usada neste cenário, tal como no anterior, é o TCP.

```
core@xubuncore:~$ ffmpeg -i video2_180_120_200k.mp4
ffmpeg version 4.2.4-1ubuntu0.1 Copyright (c) 2000-2020 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.3.0-10ubuntu2)
  configuration: --prefix=/usr --extra-version=1ubuntu0.1 --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 --e
nable-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-libfribid
i --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libo
penmpt --enable-libopus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine --enable-libsnappy --ena
ble-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 --en
able-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-libdc1394 --en
able-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 'video2_180_120_200k.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.29.100
  Duration: 00:00:15.57, start: 0.000000, bitrate: 162 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 180x120, 159 kb/s, 30 fps, 30 tbr, 15360 tbn, 60 tbc
(default)
    Metadata:
      handler_name    : VideoHandler
```

Figura 5: Bitrate necessária para o *streaming* do video 2

3 Ajuste o débito dos links da topologia de modo que o cliente no portátil 2 exiba o vídeo de menor resolução e o cliente no portátil 1 exiba o vídeo com mais resolução. Mostre evidências.

Para o portátil 2 exibir o vídeo de menor resolução, limitou-se a largura de banda do link de acesso ao portátil para um valor ligeiramente superior ao *bitrate* desse vídeo. Quanto ao portátil 1, não é necessário limitar a largura de banda pois o DASH irá utilizar o de maior qualidade (caso a conexão o permita). [Figura 6]

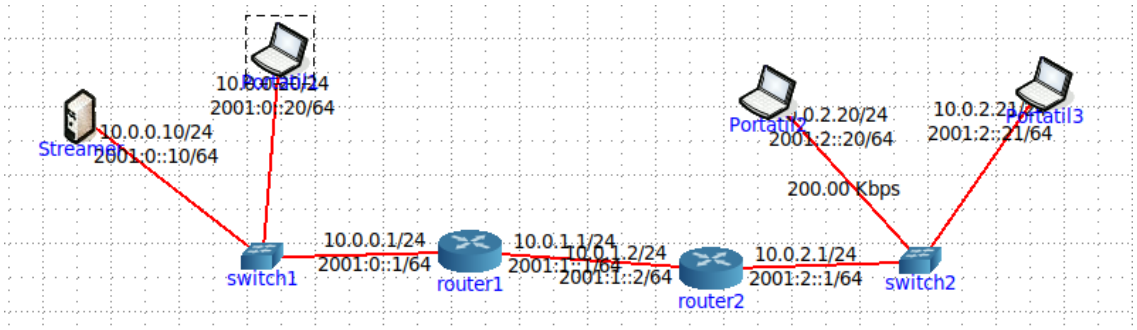
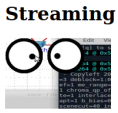


Figura 6: Topologia com o link de acesso ao portátil 2 com largura de banda limitada

Com a execução desta topologia esperávamos que ao executar o *firefox* no portátil 2, apenas fosse exibido o vídeo de menor resolução no entanto, verificou-se uma situação atípica em que tal não acontecia.

Após nova tentativa realizada a partir de casa em vez de na universidade, conseguimos obter os resultados esperados. [Figura 7]



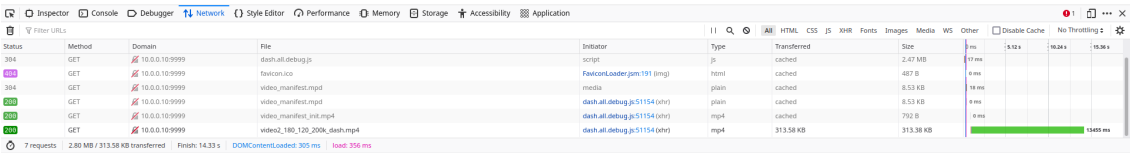


Figura 7: Topologia com o link de acesso ao portátil 2 com largura de banda limitada

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

O DASH é uma ferramenta adaptativa de *streaming* de iniciativa da DASH *Industry Forum* para estabelecer um produto de qualidade para reprodutores de vídeo e áudio, no formato *MPEG-DASH*. Por outras palavras, isto significa que é permitida à *stream* trocar de *bitrate* tendo em conta a largura de banda disponível, de maneira ao vídeo continuar a reproduzir(não ficar em *buffering*). O ficheiro MPD é um ficheiro xml com informação sobre as várias *streams* e a largura de banda associada a cada uma delas , como também outros metadados como os *codecs* a usar e como o tipo MIME. Na página HTML é referenciado este ficheiro MPD e não os ficheiros multimédia, como acontece no caso de não usarmos técnicas adaptativas. O *browser* tendo em conta a largura de banda que tem disponível escolhe a melhor *stream*, dentro das possibilidades, que melhor consegue reproduzir.

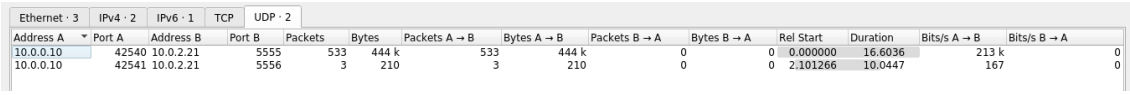
Neste caso, como o débito máximo para o portátil 2 é igual a 200kbps e o vídeo de menor resolução necessita de 162kbps, o browser irá adaptar-se e exibir este vídeo como única opção disponível.

5 Compare o cenário *unicast* aplicado com o cenário *multicast*. Mostre vantagens e desvantagens na solução *multicast* ao nível da rede, no que diz respeito a escalabilidade (aumento do nº de clientes) e tráfego na rede. Tire as suas conclusões.

No cenário *unicast*, a *stream* envia pacotes para um único nodo da rede. Caso se pretenda fazer *streaming* de um para um esta é a melhor solução pois uma das vantagens do *unicast* é que permite que os *endpoints* recebam vídeo com base no dispositivo a ser servido e na largura de banda disponível.

As soluções *multicast* assumem que todos os clientes têm a mesma largura de banda e que estão a usar um dispositivo similar, dificultando assim a escalabilidade do número de clientes, pois com o aumento do

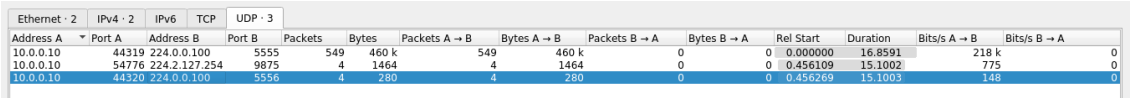
número de clientes aumenta também a probabilidade de haver flutuações na largura de banda ou diferentes tipos de dispositivos para os quais a *stream* se destina. Geralmente soluções *multicast* são usadas em LAN. Isto traz benefícios de segurança pois não estamos expostos à *internet* e por consequência não estamos tão expostos, o que nos oferece uma camada de segurança à *stream*. Como o *multicast* é *one-to-many streaming*, isto é, temos uma *stream* que é enviada de um ponto e recebido em vários *end-points*, conseguimos prever a largura de banda necessária e obter uma melhor gestão da mesma. Além disso, não importa o número de pessoas a ver a *stream* que a largura de banda é uniforme, pois todos os *end-points* recebem a mesma *stream*, não podendo assim se tratar de uma *stream* adaptativa.



Ethernet · 3 | IPv4 · 2 | IPv6 · 1 | TCP | UDP · 2

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.10	42540	10.0.2.21	5555	533	444 k	533	444 k	0	0	0.000000	16.6036	213 k	0
10.0.0.10	42541	10.0.2.21	5556	3	210	3	210	0	0	2.101266	10.0447	167	0

Figura 8: Dados referentes ao *streaming* do video 1 com *unicast*



Ethernet · 2 | IPv4 · 2 | IPv6 | TCP | UDP · 3

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.10	44319	224.0.0.100	5555	549	460 k	549	460 k	0	0	0.000000	16.8591	218 k	0
10.0.0.10	54776	224.2.127.254	9875	4	1464	4	1464	0	0	0.456109	15.1002	775	0
10.0.0.10	44320	224.0.0.100	5556	4	280	4	280	0	0	0.456269	15.1003	148	0

Figura 9: Dados referentes ao *streaming* do video 1 com *multicast*

6 Conclusão

Com este trabalho como estávamos à espera conseguimos verificar que o uso do UDP para *streaming* é mais eficiente do que o uso do TCP uma vez que é mais rápido. Relativamente ao TCP o uso de DASH em detrimento do HTTP simples sem adaptação dinâmica de débito permite que a *stream* mude a qualidade do vídeo de modo a adaptar-se à *internet* do cliente de forma a que este não tenha que esperar pelo *buffering*. No que toca ao UDP tivemos oportunidade de testar uma versão *Unicast* e outra *Multicast* e observar as vantagens e desvantagens de cada uma em relação à outra.