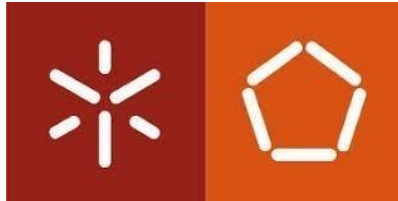


Universidade do Minho

Mestrado em Engenharia Informática



Engenharia de Serviços em Redes

1.º Ano, 1.º Semestre

Ano letivo 2021/2022

Trabalho Prático 2

Novembro 2021

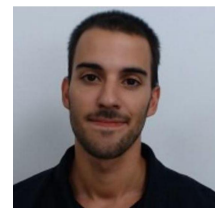
Grupo PL68



Pedro Oliveira – PG47570



Luis Pereira – PG47424



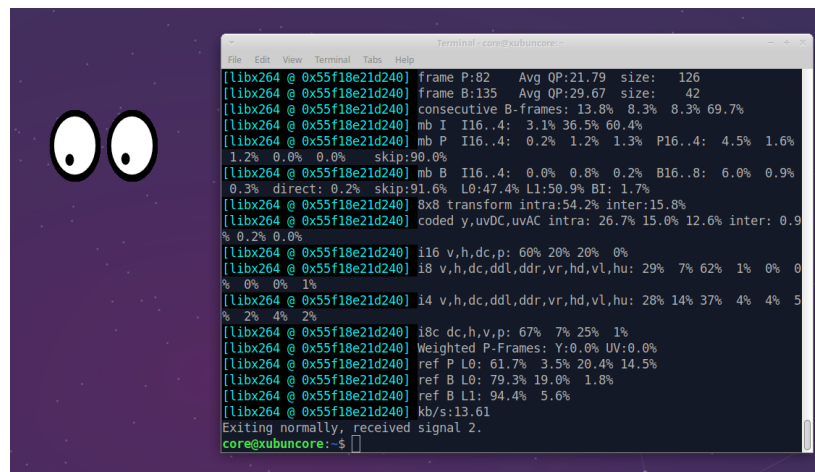
Guilherme Martins - A70782

1. Questões e Respostas

- 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 ffmpeg). 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).

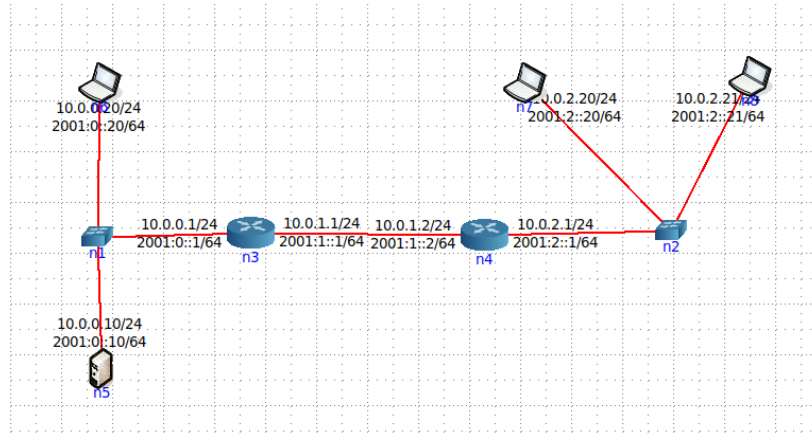
Com a ferramenta de linha de comando FFMPEG, podemos converter formatos de vídeo e áudio, bem como capturar e codificar em *real-time* a partir de várias fontes de *hardware* e *software*.

Nesta primeira etapa, utilizamos o `ffmpeg` para capturar dois pequenos vídeos (.MP4) da aplicação gráfica *xeyes*, com dimensões específicas.



```
File Edit View Terminal Tabs Help
[libx264 @ 0x55f18e21d240] frame P:82 Avg OP:21.79 size: 126
[libx264 @ 0x55f18e21d240] frame B:135 Avg OP:29.67 size: 42
[libx264 @ 0x55f18e21d240] consecutive B-frames: 13.8% 8.3% 8.3% 69.7%
[libx264 @ 0x55f18e21d240] mb I I16..4: 3.1% 36.5% 60.4%
[libx264 @ 0x55f18e21d240] mb P I16..4: 0.2% 1.2% 1.3% P16..4: 4.5% 1.6%
1.2% 0.0% 0.0% skip:90.0%
[libx264 @ 0x55f18e21d240] mb B I16..4: 0.0% 0.8% 0.2% B16..8: 6.0% 0.9%
0.3% direct: 0.2% skip:91.6% I0:47.4% I1:50.9% BI: 1.7%
[libx264 @ 0x55f18e21d240] 8x8 transform intra:54.2% inter:15.8%
[libx264 @ 0x55f18e21d240] coded y,uvDC,uvAC intra: 26.7% 15.0% 12.6% inter: 0.9%
0.2% 0.0%
[libx264 @ 0x55f18e21d240] i16 v,h,dc,p: 60% 20% 20% 0%
[libx264 @ 0x55f18e21d240] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 29% 7% 62% 1% 0% 0%
[libx264 @ 0x55f18e21d240] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 28% 14% 37% 4% 4% 5%
[libx264 @ 0x55f18e21d240] i8c dc,h,v,p: 67% 7% 25% 1%
[libx264 @ 0x55f18e21d240] Weighted P-Frames: Y:0.0% UV:0.0%
[libx264 @ 0x55f18e21d240] ref P l0: 61.7% 3.5% 20.4% 14.5%
[libx264 @ 0x55f18e21d240] ref B l0: 79.3% 19.0% 1.8%
[libx264 @ 0x55f18e21d240] ref B l1: 94.4% 5.6%
[libx264 @ 0x55f18e21d240] kb/s:13.61
Exiting normally, received signal 2.
core@xubuncore:~$
```

De seguida, procedemos a criar uma topologia CORE para fazer *streaming* do video2.mp4 de um servidor para diferentes máquinas.



Com recurso ao Wireshark, é possível capturar o tráfego para diferentes cenários de *streaming*:

Com 1 cliente (VLC):

7	0.054868824	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[ACK]	Seq=4000 Ack=1 Win=509 Len=1448 TSval=4947...
8	0.054869313	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[ACK]	Seq=5448 Ack=1 Win=509 Len=1448 TSval=4947...
9	0.054869446	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[ACK]	Seq=6896 Ack=1 Win=509 Len=1448 TSval=4947...
10	0.054869541	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[ACK]	Seq=8344 Ack=1 Win=509 Len=1448 TSval=4947...
11	0.054869638	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[PSH, ACK]	Seq=9792 Ack=1 Win=509 Len=1448 TSval=...
12	0.054899898	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=5448 Win=701 Len=0 TSval=3039796...
13	0.054901236	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=6896 Win=695 Len=0 TSval=3039796...
14	0.054901855	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=8344 Win=690 Len=0 TSval=3039796...
15	0.054902496	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=9792 Win=684 Len=0 TSval=3039796...
16	0.054903099	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=11240 Win=678 Len=0 TSval=303979...
17	0.054910228	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096	[ACK]	Seq=11240 Ack=1 Win=509 Len=1448 TSval=494...
18	0.054910374	10.0.0.10	10.0.0.20	TCP	1195	8080 → 47096	[PSH, ACK]	Seq=12688 Ack=1 Win=509 Len=1129 TSva...
19	0.054916992	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=12688 Win=673 Len=0 TSval=303979...
20	0.054917661	10.0.0.20	10.0.0.10	TCP	66	47096 → 8080	[ACK]	Seq=1 Ack=13817 Win=668 Len=0 TSval=303979...

Com 2 clientes (VLC e Firefox):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096 [ACK] Seq=1 Ack=1 Win=509 Len=1448 TSval=4949037...
2	0.000000700	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096 [ACK] Seq=1449 Ack=1 Win=509 Len=1448 TSval=4949...
3	0.000000950	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096 [ACK] Seq=2897 Ack=1 Win=509 Len=1448 TSval=4949...
4	0.000001140	10.0.0.10	10.0.0.20	TCP	611	8080 → 47096 [PSH, ACK] Seq=4345 Ack=1 Win=509 Len=545 TSval=...
9	0.000139401	10.0.0.10	10.0.2.20	TCP	1514	8080 → 59818 [ACK] Seq=1 Ack=1 Win=507 Len=1448 TSval=2797589...
10	0.000139687	10.0.0.10	10.0.2.20	TCP	1514	8080 → 59818 [ACK] Seq=1449 Ack=1 Win=507 Len=1448 TSval=2797...
11	0.000139940	10.0.0.10	10.0.2.20	TCP	1514	8080 → 59818 [ACK] Seq=2897 Ack=1 Win=507 Len=1448 TSval=2797...
12	0.000140165	10.0.0.10	10.0.2.20	TCP	611	8080 → 59818 [PSH, ACK] Seq=4345 Ack=1 Win=507 Len=545 TSval=...
17	0.250813280	10.0.0.10	10.0.0.20	TCP	1514	8080 → 47096 [ACK] Seq=4890 Ack=1 Win=509 Len=1448 TSval=4949...

Com 3 clientes (VLC, Firefox e ffmpeg):

No.	Time	Source	Destination	Protocol	Length	Info
61	0.492342469	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=13817 Ack=1 Win=509 Len=1448 TSval=309...
62	0.492342641	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=15265 Ack=1 Win=509 Len=1448 TSval=309...
63	0.492342766	10.0.0.10	10.0.0.20	TCP	1477	8080 → 54928 [PSH, ACK] Seq=16713 Ack=1 Win=509 Len=1411 TSva...
67	0.492400194	10.0.0.10	10.0.2.20	TCP	1514	8080 → 46506 [ACK] Seq=13817 Ack=1 Win=507 Len=1448 TSval=703...
68	0.492400396	10.0.0.10	10.0.2.20	TCP	1514	8080 → 46506 [ACK] Seq=15265 Ack=1 Win=507 Len=1448 TSval=703...
69	0.492400493	10.0.0.10	10.0.2.20	TCP	1477	8080 → 46506 [PSH, ACK] Seq=16713 Ack=1 Win=507 Len=1411 TSva...
73	0.508920839	10.0.0.10	10.0.2.21	TCP	66	[TCP Keep-Alive] 8080 → 59110 [ACK] Seq=18784 Ack=1 Win=509 L...
75	0.997106691	10.0.0.10	10.0.2.21	TCP	66	[TCP Keep-Alive] 8080 → 59110 [ACK] Seq=18784 Ack=1 Win=509 L...
76	1.096454218	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=18124 Ack=1 Win=509 Len=1448 TSval=309...
77	1.096454748	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=19572 Ack=1 Win=509 Len=1448 TSval=309...
78	1.096454852	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=21020 Ack=1 Win=509 Len=1448 TSval=309...
79	1.096454948	10.0.0.10	10.0.0.20	TCP	291	8080 → 54928 [PSH, ACK] Seq=22468 Ack=1 Win=509 Len=225 TSval=...
84	1.096559659	10.0.0.10	10.0.2.20	TCP	1514	8080 → 46506 [ACK] Seq=18124 Ack=1 Win=507 Len=1448 TSval=703...
85	1.096559913	10.0.0.10	10.0.2.20	TCP	1514	8080 → 46506 [ACK] Seq=19572 Ack=1 Win=507 Len=1448 TSval=703...
86	1.096560021	10.0.0.10	10.0.2.20	TCP	1514	8080 → 46506 [ACK] Seq=21020 Ack=1 Win=507 Len=1448 TSval=703...
87	1.096560122	10.0.0.10	10.0.2.20	TCP	291	8080 → 46506 [PSH, ACK] Seq=22468 Ack=1 Win=507 Len=225 TSval=...
92	1.595499362	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=22693 Ack=1 Win=509 Len=1448 TSval=309...
93	1.595499862	10.0.0.10	10.0.0.20	TCP	1514	8080 → 54928 [ACK] Seq=24141 Ack=1 Win=509 Len=1448 TSval=309...

Taxa em bps necessária:

Verificando a *protocol hierarchy* no *Wireshark* podemos concluir que a taxa em bps é de 78k.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	402	100.0	311548	377 k	0	0	0
Ethernet	100.0	402	1.8	5628	6823	0	0	0
Internet Protocol Version 4	100.0	402	2.6	8040	9748	0	0	0
Transmission Control Protocol	99.3	399	95.6	297748	361 k	352	232506	281 k
Hypertext Transfer Protocol	11.7	47	20.7	64400	78 k	47	64400	78 k
Open Shortest Path First	0.7	3	0.0	132	160	3	132	160

Encapsulamento usado:

Analisando a imagem de cima concluímos também que foi utilizado o Ethernet, TCP e IPV4. O HTTP só é reconhecido quando é enviado o header HTTP.

Número total de fluxos gerados:

Foi aberto um fluxo por cada portátil, é verificado analisando o número de portas na conexão entre *streamer* e portátil.

tcp.stream eq 0						
No.	Time	Source	Destination	Protocol	Length	Info
2	0.000019344	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=1 Ack=1 Win=509 Len=1448 TSval=8361277...
3	0.000019586	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=1449 Ack=1 Win=509 Len=1448 TSval=...
4	0.000031546	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=2897 Ack=1 Win=509 Len=1448 TSval=8361...
5	0.000031678	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=4345 Ack=1 Win=509 Len=1448 TSval=...
6	0.000062260	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=5793 Ack=1 Win=509 Len=1448 TSval=8361...
7	0.000062426	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=7241 Ack=1 Win=509 Len=1448 TSval=...
8	0.000064904	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=8689 Ack=1 Win=509 Len=1448 TSval=8361...
9	0.000065017	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=10137 Ack=1 Win=509 Len=1448 TSva...
10	0.000081136	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=11585 Ack=1 Win=509 Len=1448 TSval=836...
11	0.000081266	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=13033 Ack=1 Win=509 Len=1448 TSva...
12	0.000083424	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=14481 Ack=1 Win=509 Len=1448 TSval=836...
13	0.000083536	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=15929 Ack=1 Win=509 Len=1448 TSva...
15	0.276945522	10.0.0.10	10.0.2.21	TCP	1474	8080 → 59110 [PSH, ACK] Seq=17377 Ack=1 Win=509 Len=1408 TSva...
73	0.508920839	10.0.0.10	10.0.2.21	TCP	66	[TCP Keep-Alive] 8080 → 59110 [ACK] Seq=18784 Ack=1 Win=509 L...
75	0.907106691	10.0.0.10	10.0.2.21	TCP	66	[TCP Keep-Alive] 8080 → 59110 [ACK] Seq=18784 Ack=1 Win=509 L...
124	1.924982561	10.0.0.10	10.0.2.21	TCP	66	[TCP Keep-Alive] 8080 → 59110 [ACK] Seq=18784 Ack=1 Win=509 L...
184	2.980784705	10.0.0.10	10.0.2.21	TCP	1298	8080 → 59110 [PSH, ACK] Seq=18785 Ack=1 Win=509 Len=1232 TSva...
185	2.980794454	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=20017 Ack=1 Win=509 Len=1448 TSval=836...
186	2.980794723	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=21465 Ack=1 Win=509 Len=1448 TSva...
187	2.980798757	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=22913 Ack=1 Win=509 Len=1448 TSval=836...
188	2.980798994	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=24361 Ack=1 Win=509 Len=1448 TSva...
189	2.980844022	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [ACK] Seq=25809 Ack=1 Win=509 Len=1448 TSval=836...
190	2.980844305	10.0.0.10	10.0.2.21	TCP	1514	8080 → 59110 [PSH, ACK] Seq=27257 Ack=1 Win=509 Len=1448 TSva...

tcp.stream eq 1						
No.	Time	Source	Destination	Protocol	Length	Info
151	2.607907399	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=41656 Win=684 Len=0 TSval=446133...
152	2.607907994	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=43104 Win=678 Len=0 TSval=446133...
154	2.607916504	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=44528 Win=673 Len=0 TSval=446133...
171	2.845777600	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=45976 Win=701 Len=0 TSval=446133...
172	2.845778652	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=47424 Win=695 Len=0 TSval=446133...
173	2.845779306	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=48872 Win=690 Len=0 TSval=446133...
174	2.845779955	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=49239 Win=688 Len=0 TSval=446133...
216	3.347992441	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=50687 Win=701 Len=0 TSval=446134...
217	3.347993833	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=52135 Win=695 Len=0 TSval=446134...
218	3.347994675	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=53583 Win=690 Len=0 TSval=446134...
219	3.347995468	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=53843 Win=689 Len=0 TSval=446134...
233	3.843398174	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=55291 Win=701 Len=0 TSval=446134...
234	3.843399182	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=56739 Win=695 Len=0 TSval=446134...
235	3.843399792	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=58187 Win=690 Len=0 TSval=446134...
236	3.843400381	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=58565 Win=688 Len=0 TSval=446134...
250	4.104856886	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=60013 Win=701 Len=0 TSval=446135...
251	4.104858387	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=61461 Win=695 Len=0 TSval=446135...
252	4.104859285	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=62909 Win=690 Len=0 TSval=446135...
253	4.104860156	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=63241 Win=688 Len=0 TSval=446135...
265	4.590787749	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=64689 Win=701 Len=0 TSval=446135...
266	4.590789194	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=66137 Win=695 Len=0 TSval=446135...
267	4.590790094	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=67555 Win=690 Len=0 TSval=446135...
280	5.008220040	10.0.0.20	10.0.0.10	TCP	66	54928 → 8080 [ACK] Seq=1 Ack=69002 Win=701 Len=0 TSval=446136...

tcp.stream eq 2									
No.	Time	Source	Destination	Protocol	Length	Info			
160	2.607990643	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=37312 Win=727 Len=0 TSval=241505...
161	2.607991724	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=38760 Win=750 Len=0 TSval=241505...
162	2.607992549	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=40208 Win=748 Len=0 TSval=241505...
163	2.607993376	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=41656 Win=742 Len=0 TSval=241505...
164	2.607994181	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=43104 Win=737 Len=0 TSval=241505...
166	2.608000941	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=44528 Win=731 Len=0 TSval=241505...
179	2.845894202	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=45976 Win=759 Len=0 TSval=241505...
180	2.845895052	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=47424 Win=753 Len=0 TSval=241505...
181	2.845895701	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=48872 Win=748 Len=0 TSval=241505...
182	2.845896349	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=49239 Win=746 Len=0 TSval=241505...
224	3.348128920	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=50687 Win=759 Len=0 TSval=241505...
225	3.348130002	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=52135 Win=753 Len=0 TSval=241505...
226	3.348130834	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=53583 Win=748 Len=0 TSval=241505...
227	3.348131639	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=53843 Win=747 Len=0 TSval=241505...
241	3.843509932	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=55291 Win=759 Len=0 TSval=241505...
242	3.843510843	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=56739 Win=753 Len=0 TSval=241505...
243	3.843511447	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=58187 Win=748 Len=0 TSval=241505...
244	3.843512063	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=58565 Win=746 Len=0 TSval=241505...
258	4.104940059	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=60013 Win=759 Len=0 TSval=241505...
259	4.104941272	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=61461 Win=753 Len=0 TSval=241505...
260	4.104942167	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=62909 Win=748 Len=0 TSval=241505...
261	4.104943042	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=63241 Win=747 Len=0 TSval=241505...
271	4.590926024	10.0.2.20	10.0.0.10	TCP	66	46506 → 8080	[ACK]	Seq=1	Ack=64680 Win=750 Len=0 TSval=241505...

Em termos de escalabilidade, reparamos que neste caso, quanto mais clientes ativos, mais lento fica o sistema e a transmissão de streamer para portáteis.

- 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.**

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	802	100.0	967303	429 k	0	0	0
Ethernet	100.0	802	1.2	11228	4984	0	0	0
Internet Protocol Version 6	0.2	2	0.0	80	35	0	0	0
Internet Protocol Version 4	99.5	798	1.6	15960	7084	0	0	0
Transmission Control Protocol	98.3	788	97.1	939467	417 k	786	938856	416 k
Hypertext Transfer Protocol	0.2	2	94.5	914235	405 k	1	337	149
MP4 / ISOBMFF file format	0.1	1	94.5	913693	405 k	1	913898	405 k
Open Shortest Path First	1.2	10	0.0	440	195	10	440	195
Address Resolution Protocol	0.2	2	0.0	56	24	2	56	24

É necessário 405 kb/s para que o cliente de streaming consiga receber o vídeo no Firefox.

A pilha protocolar usada neste cenário é a apresentada no ‘print’ acima, fazendo recurso a TCP e HTTP.

1.3. **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.**


Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	312	100.0	277229	138 k	0	0	0
Ethernet	100.0	312	1.6	4368	2181	0	0	0
Internet Protocol Version 6	0.6	2	0.0	80	39	0	0	0
Internet Protocol Version 4	99.4	310	2.2	6200	3096	0	0	0
Transmission Control Protocol	96.5	301	96.0	266113	132 k	299	265200	132 k
Hypertext Transfer Protocol	0.6	2	79.7	220945	110 k	1	336	167
MP4 / ISOBMFF file format	0.3	1	79.5	220404	110 k	1	220609	110 k
Open Shortest Path First	2.9	9	0.1	396	197	9	396	197

Streaming ERS — Mozilla Firefox (on Portatil1)

Streaming ERS

10.0.0.10:9999/video_dash.html

Streaming ERS: etapa 2 DASH




Streaming ERS — Mozilla Firefox (on Portatil2)

Streaming ERS

10.0.0.10:9999/video_dash.html

Streaming ERS: etapa 2 DASH



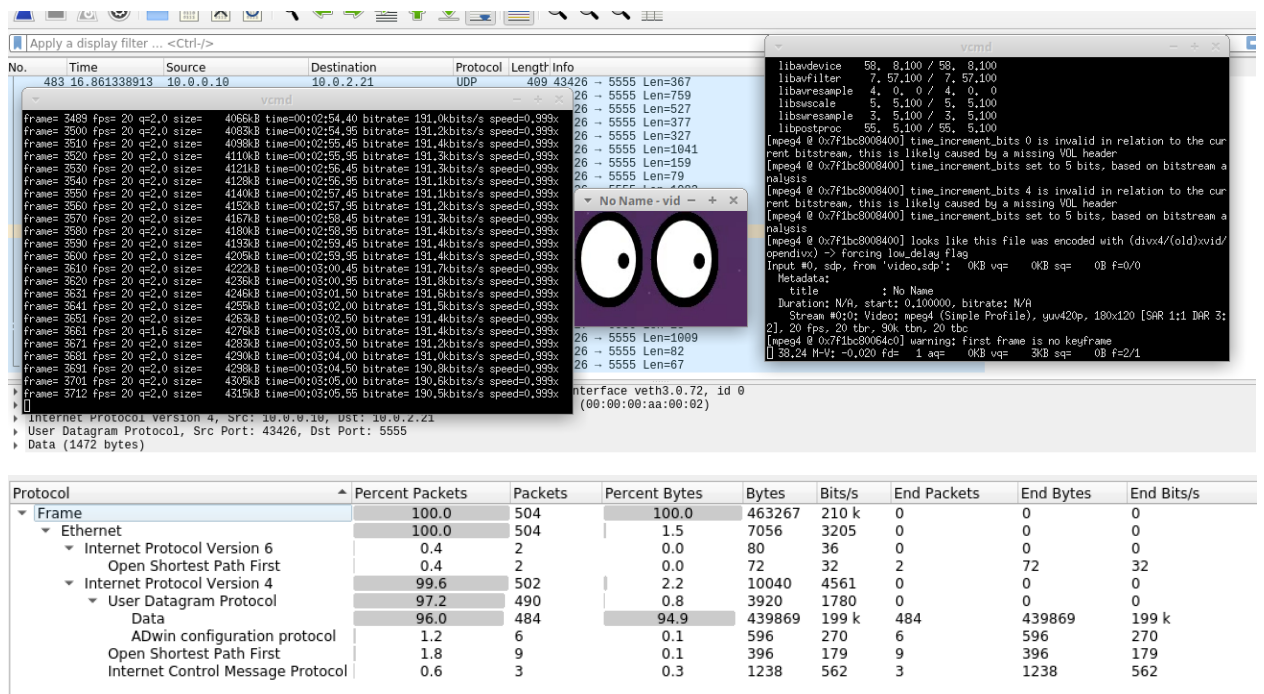
1.4. Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.

Neste caso o DASH parte os nossos vídeos 2 em vários segmentos que serão disponibilizados em diferentes *bit rates*.

O ficheiro *MPD* contém a *metadata* a ser utilizada pelo DASH cliente, em formato *xml* contendo informações sobre os segmentos e *bitrate*. Esta informação permite o streaming de um vídeo com qualidade variável dependendo da *bitrate* atual. Assim, podemos disponibilizar vídeo pela internet através de servidores de web HTTP convencionais.

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 n° de clientes) e tráfego na rede. Tire as suas conclusões.

Cenário Unicast:



Cenário Multicast:

The screenshot displays a VLC media player interface with a network packet capture (Wireshark) overlaid. The top window shows stream information for a multicast stream (Stream #0:0(und): Video: h264 (High) (avc1 / 0x31B37661), yuv420p, 180x120, 17 kb/s, 20 fps, 20 tbr, 10240 t). The bottom window shows a network packet capture (Wireshark) with a list of protocols including Ethernet, Internet Protocol Version 4, User Datagram Protocol, Session Announcement Protocol, Session Description Protocol, Real-time Transport Protocol, MP4V-ES, Real-time Transport Control Protocol, and Data. The packet list shows a Real-time Transport Protocol packet (87.0) and a Data packet (11.5).

Numa transmissão unicast, é identificado a interface do destino e são enviados IP packets do servidor para um único cliente na rede com o endereço associado a essa interface.

Para uma transmissão multicast, é identificado um grupo de interfaces e os IP packets são enviados do servidor para múltiplos clientes associados a essas interfaces.

A transmissão multicast é vantajosa em relação à unicast pois não há envio desnecessário de pacotes, enquanto que em unicast os dados são enviados para cada cliente individualmente, em multicast, o servidor envia os dados apenas uma vez, permitindo que aplicações escalem mais facilmente e que sirvam um grande número de clientes sem sobrecarregar a rede. Por outro lado, a transmissão multicast introduz mais complexidade na rede e não garante a entrega confiável dos dados ao destino pois os serviços de controle de erros, controle de fluxos e de congestionamento são tratados na camada acima, a de transporte.

2. Conclusões

Quanto à parte pedagógica, concluímos que este projeto foi muito enriquecedor para o nosso coletivo, pois este trabalho permitiu que melhorássemos as nossas competências a nível prático relacionadas com a UC de ESR.

Conseguimos aprofundar conhecimentos sobre ferramentas como o DASH, que permite ‘streaming’ de vídeos com qualidade variável fazendo recurso a servidores de ‘web’ HTTP convencionais.

Podemos também compreender as vantagens e desvantagens da utilização do unicast e do multicast, conforme as suas complexidades e eficiências.

Os elementos do grupo acreditam que a solução apresentada corresponde às expetativas, pois os resultados são plausíveis.

Em suma, o esforço coletivo foi grande com o intuito de garantir boas soluções para o enunciado proposto deixando, assim, uma janela aberta de novas ideias para trabalhos futuros.

3. Bibliografia

Dash

https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP

Dash-MPD

https://ottverse.com/structure-of-an-mpeg-dash-mpd/#DASH_MPD_Format