

Trabalho Prático 1

Engenharia de Serviços em Rede

PL34

Mariana Rodrigues^[pg50622], Jorge Melo^[pg50507], and Inês Vicente^[pg50436]

Universidade do Minho

ETAPA 1 - Streaming HTTP simples sem adaptação dinâmica de débito

Questão 1: Capture três pequenas amostras de tráfego no link de saída do servidor, respetivamente com 1 cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay). Identifique a taxa em bps necessária (usando o `ffmpeg -i videoA.mp4` e/ou o próprio wireshark), o encapsulamento usado e o número total de fluxos gerados. Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício (ex: capturas de ecrã).

Inicialmente, começamos por fazer a captura do **videoA.mp4** e do **videoB.mp4**.



Figura 1. Vídeos A e B

Prosseguindo para a construção da topologia base no CORE.

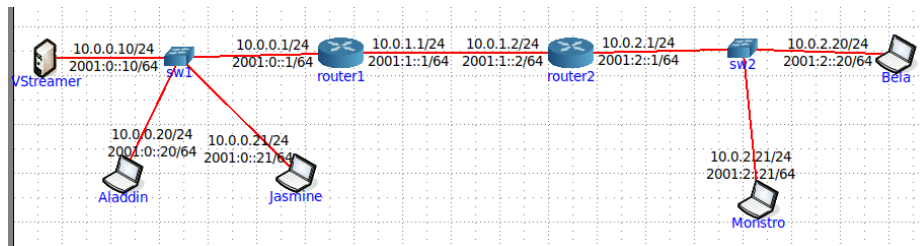
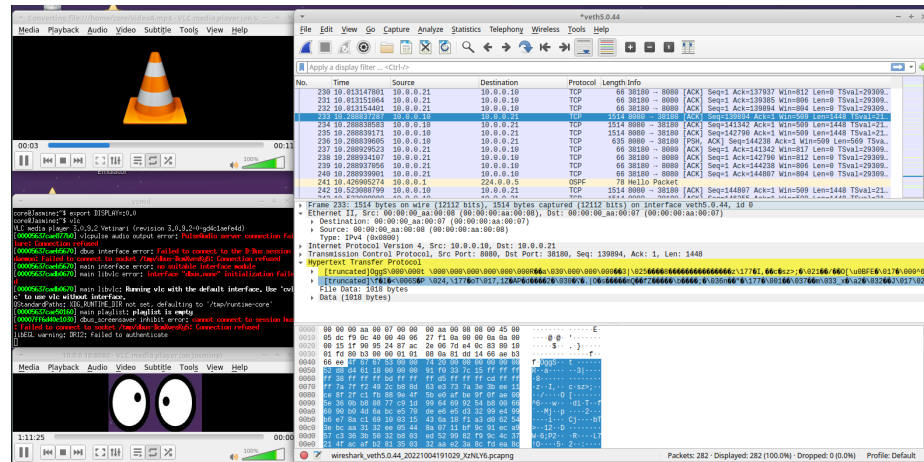


Figura 2. Topologia Core

```
vcmd
root@VStreamer:/tmp/pycore.42005/VStreamer.conf# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data:
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.105 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.122 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.134 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.052 ms
64 bytes from 10.0.2.20: icmp_seq=5 ttl=62 time=0.118 ms
64 bytes from 10.0.2.20: icmp_seq=6 ttl=62 time=0.161 ms
64 bytes from 10.0.2.20: icmp_seq=7 ttl=62 time=0.094 ms
64 bytes from 10.0.2.20: icmp_seq=8 ttl=62 time=0.133 ms
^C
--- 10.0.2.20 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7128ms
rtt min/avg/max/mdev = 0.052/0.114/0.161/0.030 ms
root@VStreamer:/tmp/pycore.42005/VStreamer.conf#
```

Figura 3. Teste à conectividade da rede

Com a topologia a funcionar e os vídeos capturados anteriormente, avançamos para fazer o streaming por **HTTP**, no **VStreamer** com o **VLC** do ficheiro *videoA.mp4* com transcoding para “**Video – Theora + Vorbis (Ogg)**”. Posteriormente, colocamos o **VLC** no portátil **Jasmine**, mas desta vez a funcionar como cliente.



Tendo isto, prosseguimos para a transmissão do videoA.mp4 através de uma página HTML.

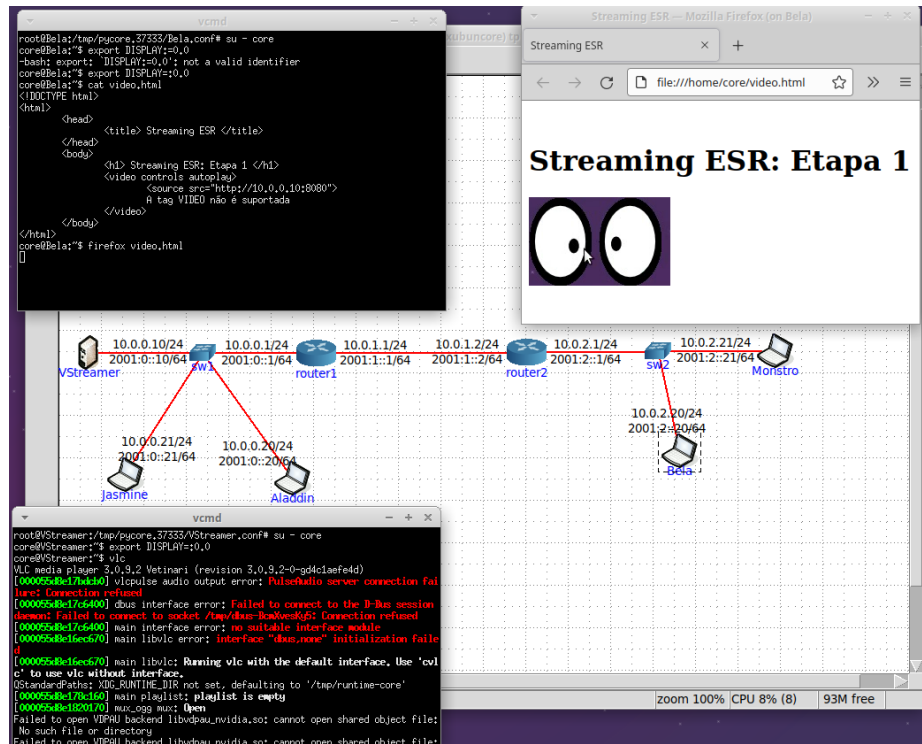


Figura 5. Verificação do funcionamento do vídeo na página do Firefox

Verificando que estava tudo a correr como previsto, efetuamos a recolha de uma amostra de tráfego.

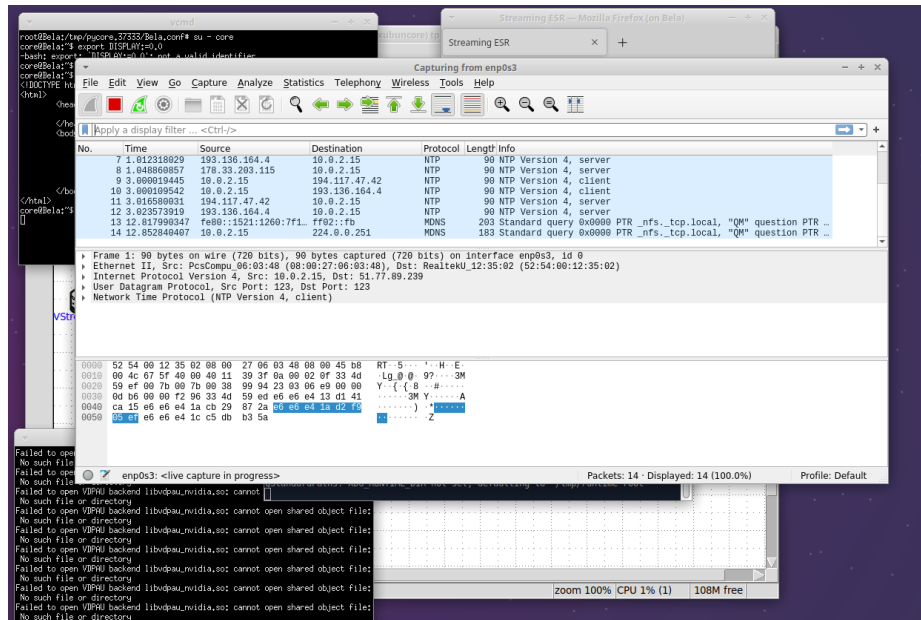


Figura 6. Recolha de uma amostra de tráfego no wireshark

Posto isto, tendo os 2 clientes anteriores a funcionar separadamente, seguimos para o teste de ambos os clientes em conjunto.

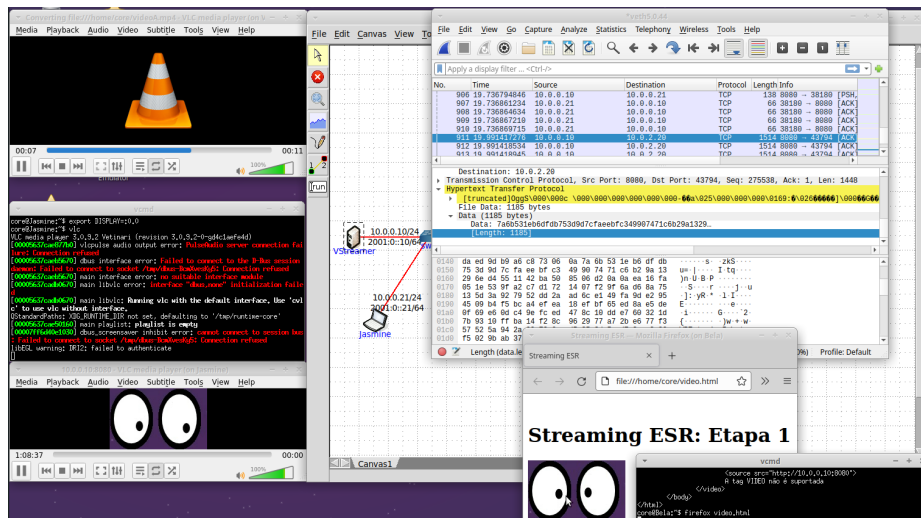


Figura 7. 2 clientes (VLC e Firefox)

Por último, efetuamos a stream de vídeo usando o comando **ffplay** como um terceiro cliente no portátil **Monstro**.

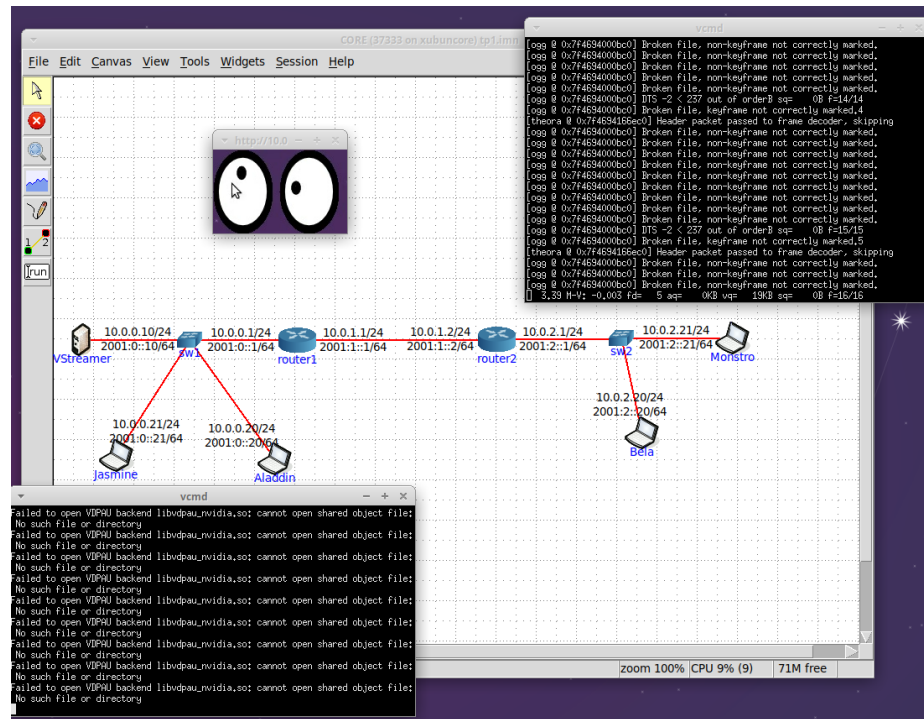


Figura 8. Três clientes (VLC, Firefox e ffplay)

Com os três clientes a correr, as imagens anteriores e com recolhas de informação a partir da ferramenta "Wireshark", verificámos que:

- **encapsulamento usado:**
 - **VLC** : Ethernet II, IPv4 e TCP
 - **Firefox** : Ethernet II, IPv4, TCP e Hypertext Transfer Protocol
 - **FFPlay** : Ethernet II, IPv4, TCP e Hypertext Transfer Protocol
- **total de fluxos gerados:**

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
10.0.0.21	58140	10.0.0.10	8080	1,005	684 k	503	33 k	502	651 k	0.000000	43.0419	6170	121 k
10.0.2.20	59658	10.0.0.10	8080	1,006	685 k	503	33 k	503	652 k	0.000134	43.0420	6170	121 k
10.0.2.21	45706	10.0.0.10	8080	1,006	685 k	503	33 k	503	652 k	0.002401	43.0411	6170	121 k

Figura 9. Informacao relativa as amostras de tragefo no link de saida do servidor

- **VLC:** 503
- **Firefox:** 503
- **ffplay:** 503
- **taxa em bps necessárias:**

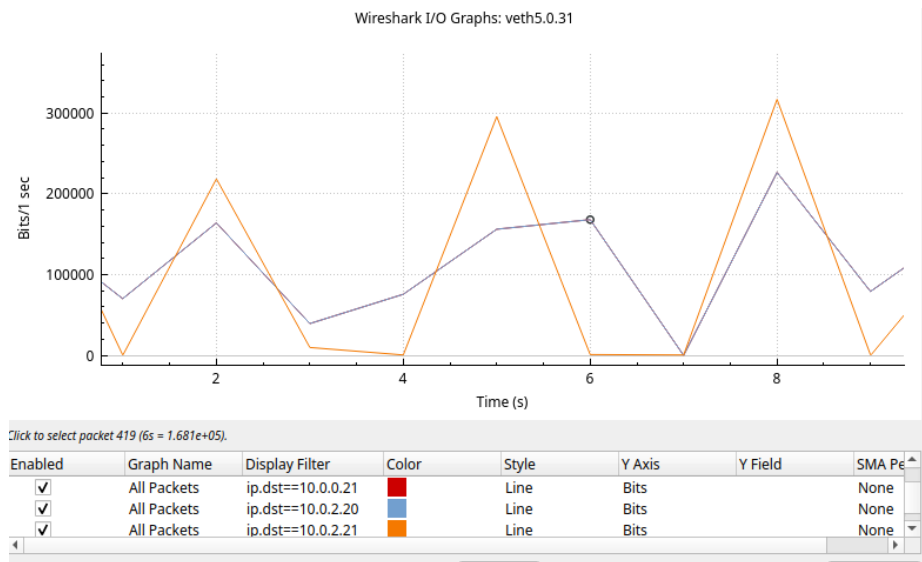


Figura 10. Wireshark I/O Graph dos bps dos 3 clientes

Etapa 2 - Streaming adaptativo sobre HTTP (MPEG-DASH)

Procedendo, agora, para a criação de três variantes do **vídeoB.mp4**, com diferentes resoluções

Começamos por colocar os vídeos a serem visualizados no firefox, em simultâneo, nos portáteis **Bela** e **Aladdin**.

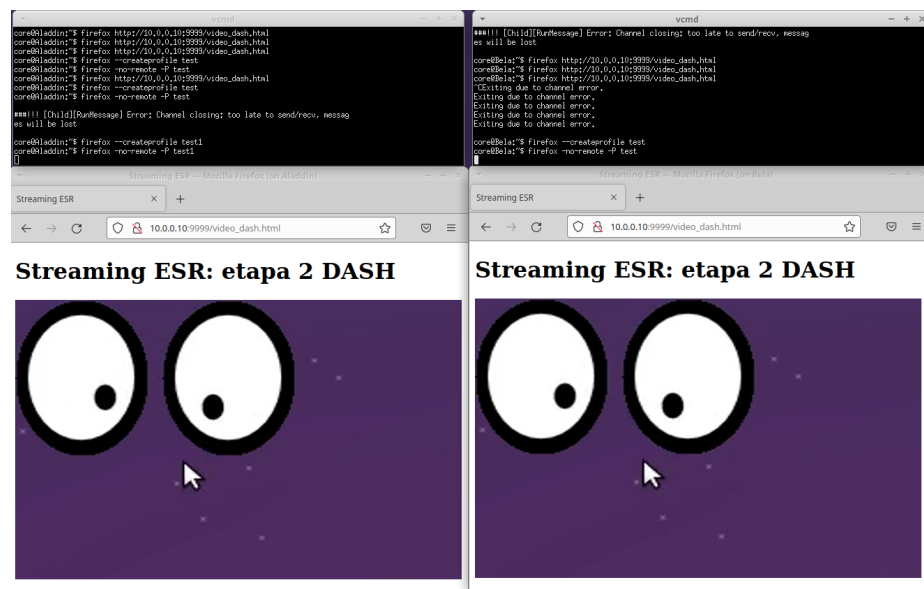


Figura 11. Visualização com o firefox nos portáteis Bela e Aladdin

No.	Time	Source	Destination	Protocol	Length	Info
881	24.236270260	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1041113 Ack=340 Win=64896 Len=1448 TSv...
882	24.236271554	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1042561 Ack=340 Win=64896 Len=1448 TSv...
883	24.236272853	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1044009 Ack=340 Win=64896 Len=1448 TSv...
884	24.236274106	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1045457 Ack=340 Win=64896 Len=1448 TSv...
885	24.236275373	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1046905 Ack=340 Win=64896 Len=1448 TSv...
886	24.236277433	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [ACK] Seq=1048353 Ack=340 Win=64896 Len=1448 TSv...
887	24.236278661	10.0.0.10	10.0.2.20	TCP	1514	9999 → 35516 [PSH, ACK] Seq=1049881 Ack=340 Win=64896 Len=144...
888	24.236308834	10.0.0.10	10.0.2.20	MP4	237	
889	24.238289469	10.0.2.20	10.0.0.10	TCP	66	35516 → 9999 [ACK] Seq=340 Ack=1051421 Win=596096 Len=0 TSv...
890	24.249894534	10.0.2.20	10.0.0.10	TCP	66	35516 → 9999 [FIN, ACK] Seq=340 Ack=1051421 Win=715136 Len=0 TSv...
891	24.250604815	10.0.0.10	10.0.2.20	TCP	66	9999 → 35516 [ACK] Seq=1051421 Ack=341 Win=64896 Len=0 TSv...
892	26.042644950	10.0.2.1	224.0.0.5	OSPF	78	Hello Packet
893	28.051945177	10.0.2.1	224.0.0.5	OSPF	78	Hello Packet

Figura 12. Captura de tráfego no portátil Bela

Questão 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.

```

<!-- MPD file generated with GPAC version 0.5.2-DEV (revision: 0.5.2-426-gc5e0e0d0f9b-5, at 2022-10-06T14:02:12.954Z) -->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H0M1.834S" maxSegmentDuration="PT0H0M0.500S" profiles="urn:mpeg:dash:profile:full:2011">
  <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
    <Title>video_manifest.mpd generated by GPAC/Title</Title>
  </ProgramInformation>
  <Period duration="PT0H0M1.834S">
    <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="640" maxHeight="480" maxFrameRate="30" par="B:5" lang="und">
      <SegmentList>
        <Initialization sourceURL="video_manifest_init.mpd"/>
      </SegmentList>
      <Representation id="1" mimeType="video/mp4" codecs="avc1.64000c" width="160" height="100" frameRate="30" sar="1:1" startWithSAP="0" bandwidth="121717">
        <BaseURL>video160_100_200k_dash.mpd</BaseURL>
        <SegmentList timescale="1250" duration="7680">
          <SegmentURL mediaRange="927-5249" indexRange="927-970"/>
          <SegmentURL mediaRange="5250-8241" indexRange="5250-5323"/>
          <SegmentURL mediaRange="8242-12259" indexRange="8242-8540"/>
          <SegmentURL mediaRange="12260-22952" indexRange="12260-12303"/>
        </SegmentList>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>

```

Figura 13. video_manifest.mpd

Observando o ficheiro *video_manifest.mpd*, podemos observar que o vídeo de menor resolução tem uma *bandwidth* de **121717** bps.

A pilha protocolar necessária para que o cliente conseguisse receber o vídeo no firefox foi com TCP, Internet Protocol e HTTP.

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

No core, alterando o limite de *bps* do link entre o Bela e o *switch* para um valor entre a *bandwidth* do vídeo com menos resolução e do segundo vídeo com menor resolução, conseguimos forçar o portátil Bela a transmitir para o firefox o vídeo com a menor resolução.

Quanto ao Alladin, o link está definido como 0, que é o valor *default*, o que significa que não há limite de tráfego que pode passar naquele link. Portanto, o vídeo de maior resolução será transmitido por defeito.

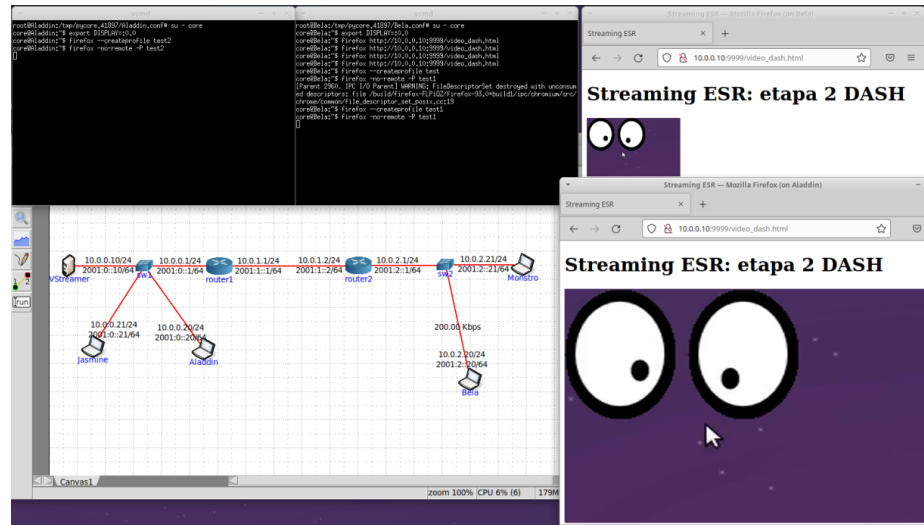


Figura 14. Visualização com o firefox no portátil Bela com o vídeo de menor resolução e do Alladin com o de maior resolução

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

O cliente tenta aceder à página. O servidor envia o ficheiro MPD, que o cliente consulta e, a partir daí, obtém a informação relativa à *bandwidth* mínima requerida pelos diferentes vídeos. Depois, tendo em conta a *bandwidth* de ligação entre o servidor e o cliente, escolhe a resolução que consegue suportar. Ou seja, ao diminuir a *bandwidth* da ligação, naturalmente o vídeo que o cliente vai obter vai ter uma menor resolução.

Etapa 3: Streaming RTP/RTCP unicast sobre UDP e multicast com anúncios SAP

Questão 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.

Começando, então, com o *unicast*, criamos uma sessão de *streaming* com RTP com o *ffmpeg* no *VStreamer* e um cliente *ffplay* no portátil *Monstro*.

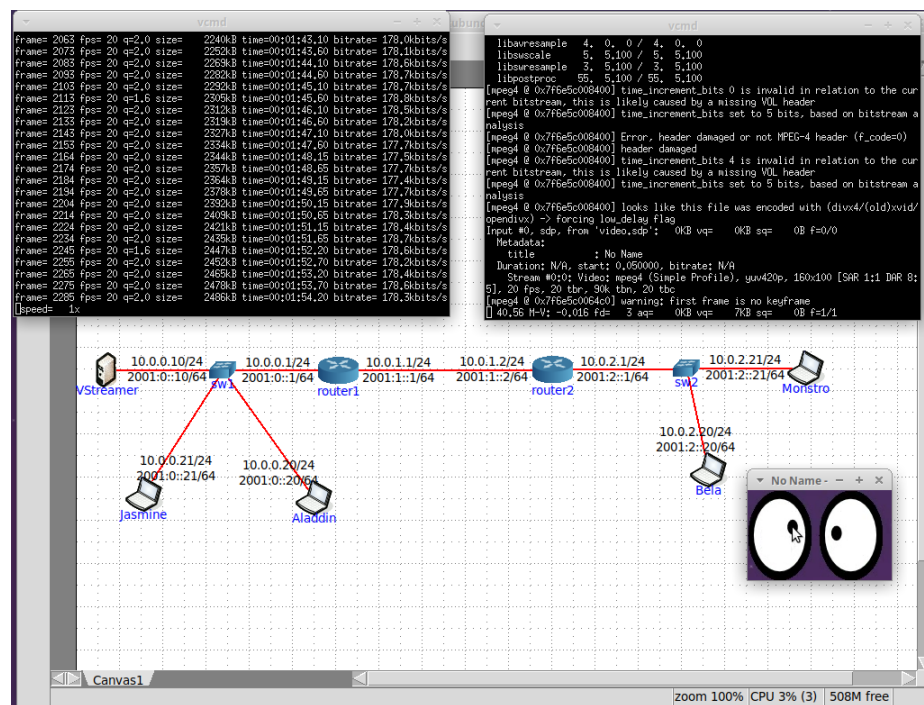


Figura 15. Sessão de *streaming unicast*

De seguida, capturámos o tráfego com o *wireshark* no link de saída do servidor.

Wireshark - Conversations - veth5.0.79

Ethernet - 3		IPv4 - 2		IPv6 - 1		TCP		UDP - 2													
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A										
10.0.0.1	224.0.0.5	5	390	5	390	0	0	0.084583	8.0180												
10.0.0.10	10.0.2.21	228	206 k	228	206 k	0	0	0.000000	8.2984												

Figura 16. Captura de tráfego para *unicast*

Podemos notar que o débito para *Unicast* é de **198k bps**. Para o *multicast*, criamos uma topologia com apenas um *switch*, com 4 portáteis e um servidor ligados ao mesmo.

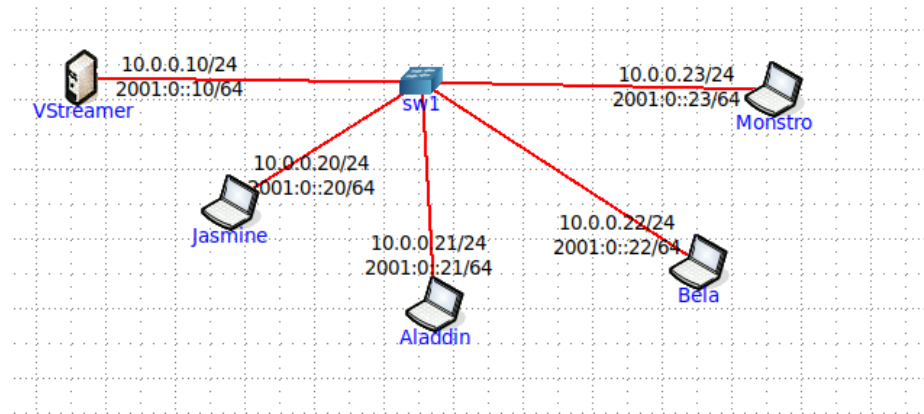


Figura 17. Topologia para *multicast*

Em seguida, iniciámos, para cada portátil, uma sessão de *streaming multicast* com o *ffmpeg* em sessões diferentes.

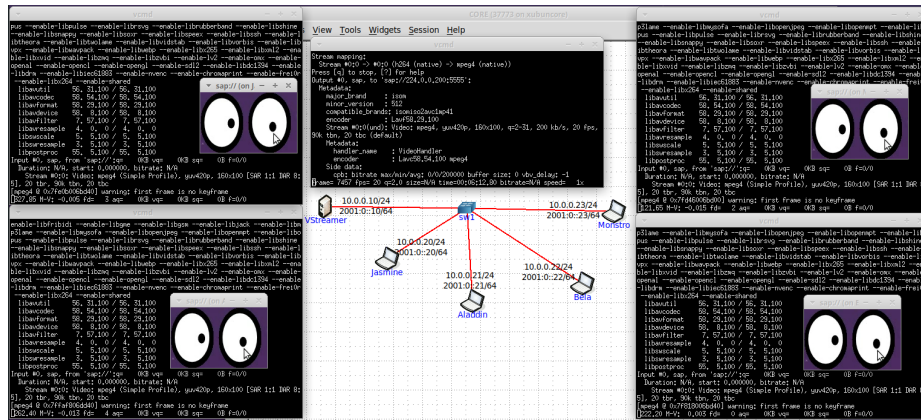


Figura 18. Vários portáteis em sessões diferentes em *multicast*

Finalmente, capturámos o tráfego no link de saída do 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
10.0.0.10	224.0.0.200	1,305	1107 k	1,305	1107 k	0	0	0.000000	46.9854	188 k	
10.0.0.10	224.2.127.254	9	3294	9	3294	0	0	1.838581	40.2915	654	

Figura 19. Captura de tráfego para *multicast*

Podemos então observar que o débito é de **188k** para *multicast*.

Para finalizar e para conseguirmos comparar ambos os cenários, fomos comparar os valores de débito:

- **unicast**: 198k
- **multicast**: 188k

Com isto, podemos concluir algumas vantagens que o *multicast* tem sobre o *unicast*. Em termos de escalabilidade, podemos observar que com um débito semelhante, o *multicast* consegue satisfazer um maior número de clientes.

Para além disso, em *unicast*, para calcular o débito total multiplica-se o número de clientes pelo débito de um cliente, enquanto que o *multicast*, que só envia o conteúdo uma vez por todas as redes, mantém um débito mais ou menos constante, pelo que aumentar o número de clientes para *unicast* é muito mais dispendioso.

Conclusão

Este projeto serviu de ferramenta para aprender vários temas abordados nesta cadeira, entre elas:

- Streaming HTTP com e sem adaptação dinâmica de débito
- funcionamento do protocolo DASH
- diferenças entre *unicast* e *multicast*, as suas vantagens e desvantagens