**Universidade do Minho**
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

# Engenharia de Serviços em Rede

Ano Letivo de 2021/2022

# Streaming de áudio e vídeo a pedido e em tempo real

João Amorim, A74806
Tiago Matos, PG45585
Gonçalo Costeira, PG47219

30 de dezembro de 2021

ERS

# Índice

# 1 Streaming HTTP simples sem adaptação dinâmica de débito

## 1.1 Tarefas

Nesta primeira Etapa, pretende-se testar o streaming sobre HTTP sem adaptação dinâmica de débito. Inicialmente, vamos começar por criar a nossa topologia, de acordo com o enunciado. A topologia irá ter um servidor, 3 portáteis, 2 switches e dois routers.

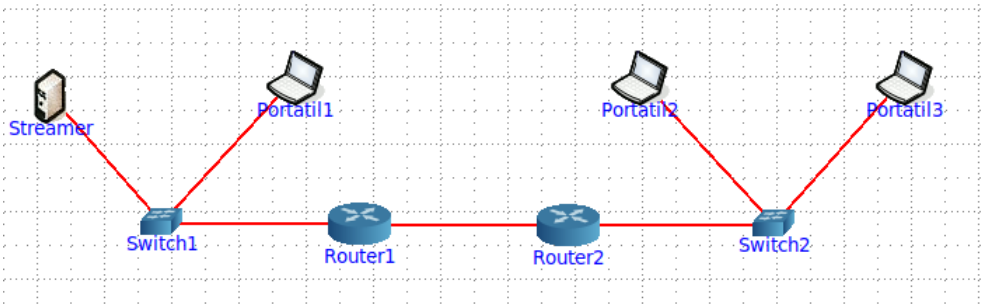


Figura 1.1: Topologia criada no CORE

De seguida, e para garantir que a topologia criada funciona sem problemas, vamos testar a conectividade da mesma, com o nmap:

```
root@Streamer:/tmp/pycore.39983/Streamer.conf# nmap -n -sP 10.0.*.*/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-08 00:34 WET
Nmap scan report for 10.0.0.1
Host is up (0.000038s latency).
MAC Address: 00:00:00:AA:00:02 (Xerox)
Nmap scan report for 10.0.0.20
Host is up (0.0000080s latency).
MAC Address: 00:00:00:AA:00:01 (Xerox)
Nmap scan report for 10.0.0.10
Host is up.
Nmap scan report for 10.0.1.1
Host is up (0.000014s latency).
Nmap scan report for 10.0.1.2
Host is up (0.000030s latency).
Nmap scan report for 10.0.2.1
Host is up (0.000029s latency).
Nmap scan report for 10.0.2.20
Host is up (0.00024s latency).
Nmap scan report for 10.0.2.21
Host is up (0.00037s latency).
```

Figura 1.2: Comando nmap -n -sP

Como podemos ver, a topologia encontra-se a funcionar tal como pretendido.

Nos próximos passos, nomeadamente do 3 ao 9, é feita uma explicação sobre como iniciar e efetuar o streaming para os vários clientes. No caso desta etapa, irá ser feito o streaming do video1.mp4, a partir de um servidor a funcionar com VLC e com o nome Streamer. Quanto aos clientes, estes têm o nome de Portatil1, Portatil2 e Portatil 3, em que o primeiro corresponde a um outro VLC, o segundo a uma pagina no Firefox e o terceiro a uma instância do ffmeg.

Ao longo da execução, serão então efetuadas 3 capturas no link de saída do servidor para os seguintes clientes:

- 1ª Captura - Portatil1 (VLC)
- 2ª Captura - Portatil1 (VLC) + Portatil2 (Firefox)
- 3ª Captura - Portatil1 (VLC) + Portatil2 (Firefox) + Portatil3 (ffmeg)

O resultado obtido foi o seguinte:

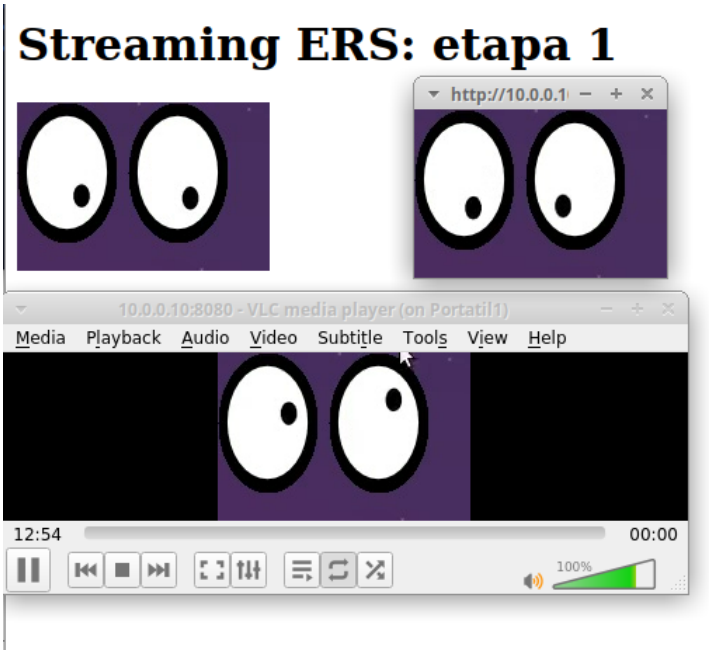


Figura 1.3: Resultado da execução das tarefas. Cliente VLC + Client Firefox + Cliente ffplay

## 1.2 Questões

### 1.2.1 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 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)

Em primeiro lugar, vamos utilizar o comando ffmpeg -i video1.mp4 para verificar a taxa em bps que é de facto necessária:

```
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: 00:00:15.75, start: 0.000000, bitrate: 17 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 180x120, 14 kb/s, 20
fps, 20 tbr, 10240 tbn, 40 tbc (default)
    Metadata:
      handler_name    : VideoHandler
```

Figura 1.4: Comando ffmpeg -i video.mp4

Tal como se vê na imagem, a taxa é de 17kb por segundo.

No que diz respeito ao encapsulamento e ao número de fluxos, recorremos às ferramentas do Wireshark. Para isto, utilizamos a "Protocol Hierarchy"em "Statistics", para identificarmos o tipo de encapsulamento usado, enquanto que para o número de fluxos, utilizamos a ferramenta "Conversations", também esta em "Statistics". Os resultados obtidos foram os seguintes:

| Protocol | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets | End Bytes |
|---|---|---|---|---|---|---|---|
| Frame | 100.0 | 1661 | 100.0 | 1134380 | 116 k | 0 | 0 |
| Ethernet | 100.0 | 1661 | 2.0 | 23254 | 2395 | 0 | 0 |
| Internet Protocol Version 6 | 0.5 | 8 | 0.0 | 320 | 32 | 0 | 0 |
| Open Shortest Path First | 0.5 | 8 | 0.0 | 288 | 29 | 8 | 288 |
| Internet Protocol Version 4 | 99.3 | 1649 | 2.9 | 32980 | 3397 | 0 | 0 |
| Transmission Control Protocol | 96.9 | 1610 | 94.8 | 1075710 | 110 k | 1455 | 855714 |
| Hypertext Transfer Protocol | 9.3 | 155 | 19.2 | 217802 | 22 k | 140 | 205600 |
| Malformed Packet | 0.9 | 15 | 0.0 | 0 | 0 | 15 | 0 |
| Open Shortest Path First | 2.3 | 39 | 0.2 | 1716 | 176 | 39 | 1716 |
| Address Resolution Protocol | 0.2 | 4 | 0.0 | 112 | 11 | 4 | 112 |

Figura 1.5: Protocol Hierarchy - Encapsulamento utilizado

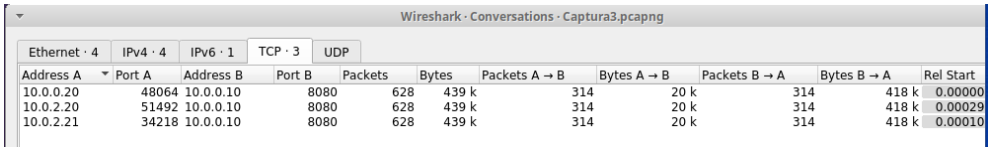


Figura 1.6: Conversations - Número de Fluxos para a Captura com os 3 Clientes em funcionamento.

Para efeito de capturas no Wireshark, todas as três capturas tiveram relativamente a mesma duração, cerca de 30 segundos, correspondendo à duração de Streaming do video1.mp4 por duas vezes. No entanto, e de acordo com as capturas efetuadas, existem atrasos no streaming do video1, para a segunda captura em relação à primeira e para a terceira captura em relação à segunda. Isto indica-nos que este atraso se iria acentuar caso fossem adicionados mais Clientes, fazendo-nos concluír que esta solução terá realmente problemas de escalabilidade.

# 2 Streaming adaptativo sobre HTTP (MPEG-DASH)

## 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.

Conforme podemos verificar nos parâmetros do ficheiro MPD, a largura de banda mínima necessária para a visualização do streaming nas várias resoluções é a seguinte

- Low resolution: 148kbps;
- Medium resolution: 382kbps;
- High resolution: 734kbps;

```
    <Title>video_manifest.mpd generated by GPAC</Title>
  </ProgramInformation>
-<Period duration="PT0H0M12.734S">
  -<AdaptationSet segmentAlignment="true" bitstreamSwitching="true"
   maxWidth="540" maxHeight="360" maxFrameRate="30" par="3:2" lang="und">
   -<SegmentList>
      <Initialization sourceURL="video_manifest_init.mp4"/>
    </SegmentList>
   +<Representation id="1" mimeType="video/mp4" codecs="avc3.64000c"
    width="180" height="120" frameRate="30" sar="1:1" startWithSAP="0"
    bandwidth="147770">
    </Representation>
   +<Representation id="2" mimeType="video/mp4" codecs="avc3.640014"
    width="360" height="240" frameRate="30" sar="1:1" startWithSAP="0"
    bandwidth="381882">
    </Representation>
   +<Representation id="3" mimeType="video/mp4" codecs="avc3.64001e"
    width="540" height="360" frameRate="30" sar="1:1" startWithSAP="0"
    bandwidth="734288">
    </Representation>
  </AdaptationSet>
```

Figura 2.1: MPD file

Em termos de pilha protocolar é utilizado HTTP, um dos protocolos mais utilizados em conteudo media, sobre TCP, um protocolo de controlo de fluxo, importante para garantir a comunicação fiável e estável, neste caso entre servidor de streaming e os clients.

## 2.2 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.

Seguindo as resoluções mínimas verificadas no exercício anterior, de forma a que o PC2 reproduza o video de menor resolução é necessário uma limitação do link entre os 148-382kbps, limitamos o link entre o PC2 e switch a 256kbps. No outro caso, PC1 reproduzir o video com maior resolução, o link tem que ter no mínimo uma largura de bada de 734kbps, configuramos uma largura de banda de 1100kbps no link entre o PC1 e o switch. Esta limitação poderia ser feita entre Routers e/ou switches, mas iria causar um bottleneck afetando os vários PCs/clients da rede.

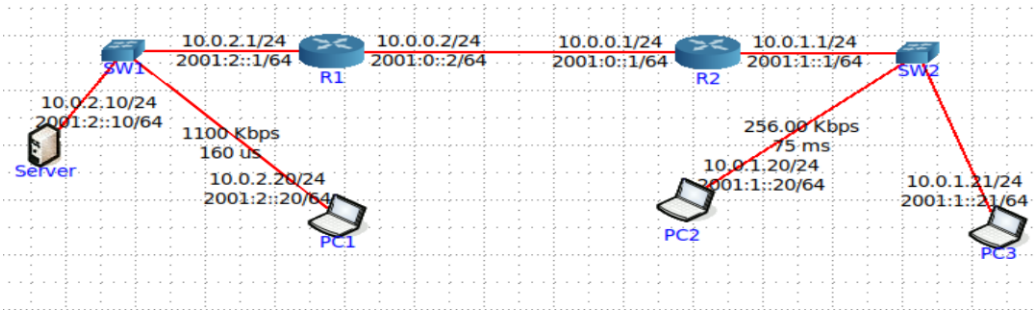


Figura 2.2: Topologia da Rede

Como podemos visualizar na captura de wireshark, no PC1 confirmamos que o video a ser reproduzido é o de menor resolução e no PC2 o de maior resolução.

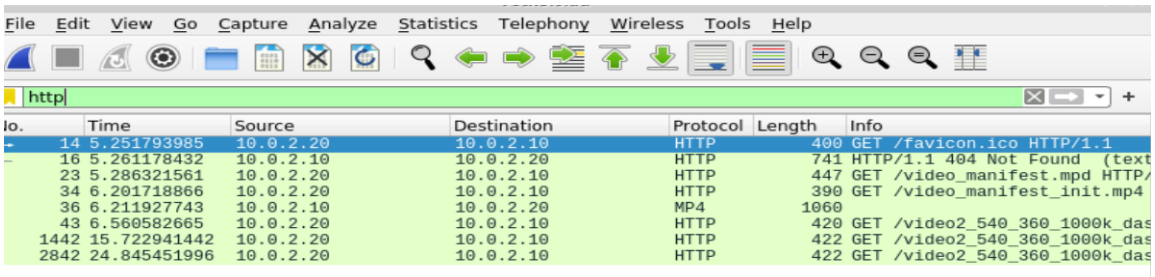


Figura 2.3: Streaming video PC1

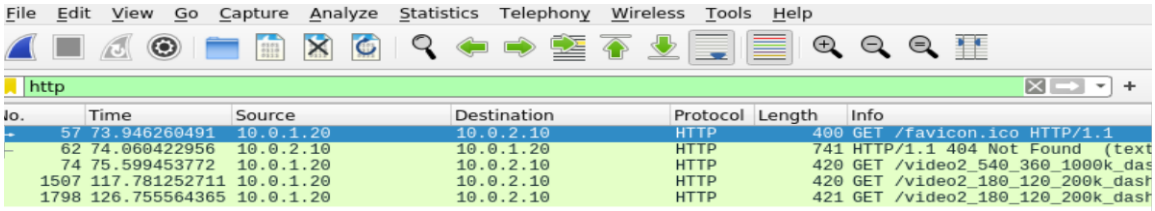


Figura 2.4: Streaming video PC2

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

Neste exercício verificamos o correto funcionamento do DASH (Dynamic Adaptive Streaming over HTTP), onde a resolução do streaming no cliente adapta-se consoante a largura de banda disponível até ao servidor. O servidor de streaming tem disponível várias versões com diferentes resoluções ou bitrates, que são descritas no manifest file MPD. O MPD file é downloaded pelo cliente, na primeira tentativa o servidor de streaming envia o vídeo com melhor resolução, caso a largura de banda permita, como no caso do PC1, ele continua a enviar o streaming com maior resolução, caso a largura de banda não permita, exemplo PC2, através do DASH existe a adaptação para o vídeo com a melhor resolução possível conforme descrito no manifest file.

# 3 Streaming RTP/RTCP unicast sobre UDP e multicast com anúnicos SAP

## 3.1 Tarefas

Nesta última etapa o streaming será sempre feito sobre UDP, quer em unicast, quer em multicast. No primeiro caso, o cliente recebe dados sobre a sessão de streaming num ficheiro de descrição em formato SDP (Session Description Protocol). O servidor ao iniciar a stream, gera o ficheiro, que deveria ser depois fornecido ao cliente, mas que neste caso não é necessário, pois o sistema de ficheiros é partilhado. O cliente com base nos dados contidos no ficheiro, inicia a receção em UDP.

Já no caso multicast o modo de funcionamento é distinto. Também é gerada uma descrição da sessão, mas ela é enviada por SAP (Session Annonucement Protocol) para um grupo multicast (endereço de grupo) especial para o efeito que é o 224.2.127.254 (sap.mcast.net) para destinos IPv4 ou ff0e::2:7ffe para destinos IPv6. Tudo o que o cliente tem de fazer é estar à escuta nesse grupo e ouvir o anúncio com os dados SDP para poder iniciar a sessão como cliente. No caso multicast, vamos escolher também um endereço de envio especial, que é um endereço de grupo. Neste exemplo foi escolhido o 224.0.0.200 porta 5555.

Para responder às questões foram realizadas uma série de tarefas para recolher a informação necessária. Para o streaming unicast podemos ver nas figuras 4.1, 4.2 e 4.3 em anexo. Já para o streaming multicast podemos ver nas figuras 4.4 e 4.5 em anexo.

## 3.2 Questões

### 3.2.1 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.

A transmissão multicast é um método de transmissão 1 para N entre um servidor e N clientes. O servidor multicast envia uma única stream, permitindo que vários clientes captem. Não há relação direta entre os clientes e o servidor. Os clientes subscrevem um grupo multicast e a rede garante a entrega da stream. Desta forma, não há sobrecarga adicional no servidor multicast se um novo cliente subscrever o grupo multicast. O servidor terá a mesma carga quer esteja 1 ou 10 000 clientes.

Unicast é uma conexão end to end. Como o servidor unicast tem uma ligação direta com cada cliente, cada novo cliente irá ocupar mais largura de banda. Devido ao aumento do consumo de rede, o unicast não é adequado para aplicações em que vários clientes estão a receber o mesmo conteúdo simultaneamente.

Como podemos ver nas figuras 4.6 e 4.7 práticamente não existe redução na velocidade de trasmissão. E são enviados exatamente os mesmos packages por segundo como seria de esperar. Então podemos concluir que a nível de escalabilidade uma solução multicast seria o mais apropriado.

# 4 Conclusões

Com este trabalho conseguimos então perceber as várias solução de streaming solicitado ou em real-time, as diferenças conceptuais entre as várias opções disponível em termos de pilhas protocolares sendo elas desde HTTP sobre TCP e ou RTP sobre UDP. Aprendemos também a utilizar várias ferramentas open-source tais como, CORE Emulator, Wireshark, FFmpeg, VideoLAN VLC e OBS Studio.

No primeiro ponto, onde era pretendido testar o streaming sobre HTTP sem DASH, verificamos que há medida que aumentamos o numero de clientes a visualizar o streaming, o delay dos streamings nos vários clientes vai aumento, verificando assim que a solução não é escalãvel.

Relativamente ao segundo ponto sobre o DASH, verificamos como é o seu funcionamento, como é efetuada a criação do conteudo em diferentes resoluções/bitrates no servidor de streaming, assim como a criação do ficheiro MPD onde podemos verificar a descrição de cada vídeo de modo que o cliente saiba quais as resoluções existentes. Verificamos o comportamento nos vários clientes com a adaptação dinamica do streaming dependendo da largura de banda disponivel entre os PCs e o servidor.

No ultimo ponto, Streaming unicast RTP/RTCP e streaming multicast, ambos sobre UDP, verificamos que no caso de unicast como é uma conexão end-to-end, o consumo de largura de banda irá ser tanto maior quando maior for o número de clientes. No caso do multicast, como é uma ligação do servidor para N clientes, o servidor terá a mesma carga. A nível de escalabilidade, obviamente a solução unicast é a mais apropriada.

# Referências

[Kurose and Ross, 2021] Kurose, J. F. and Ross, K. W. (2021). Computer networking, a top-down approach 7th.

# Anexos

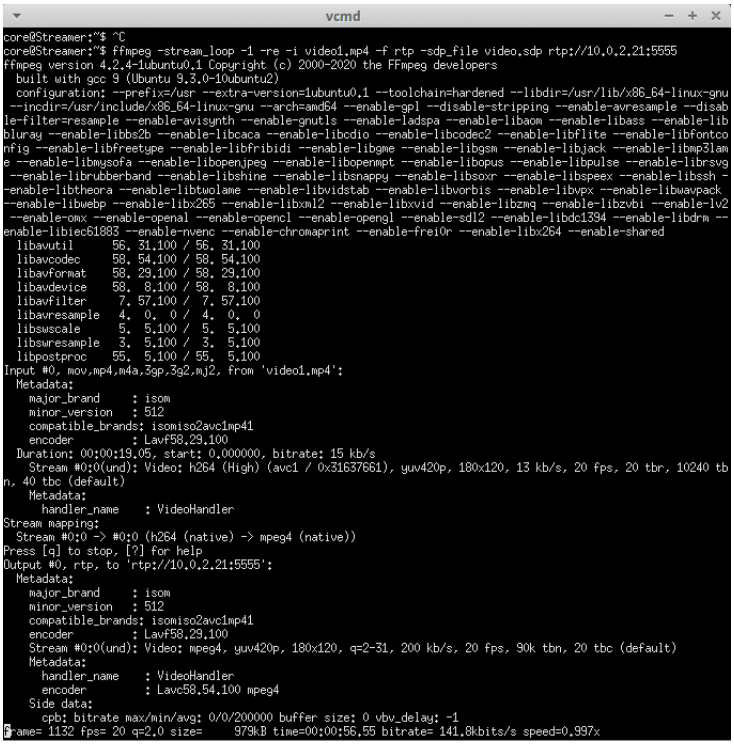


Figura 4.1: Inicio da sessão de streaming unicast com RTP no streamer

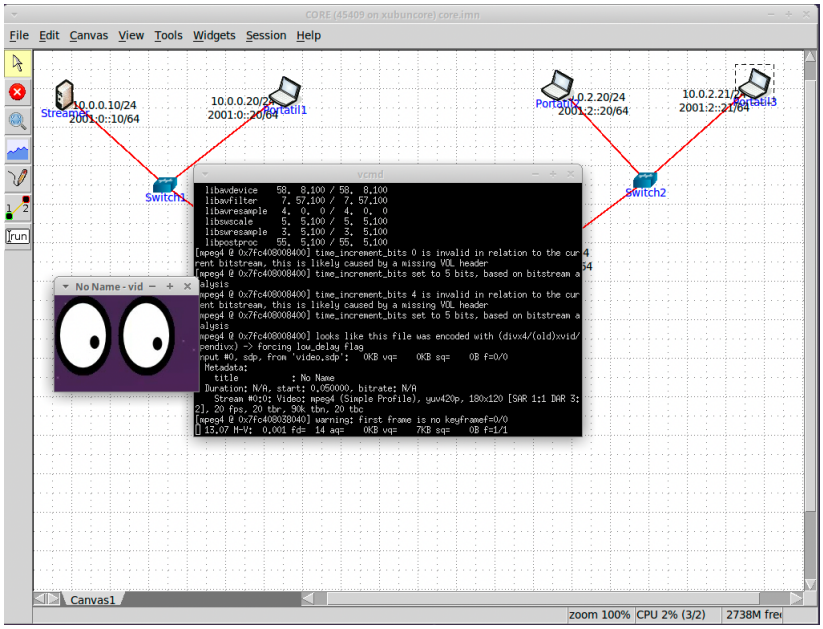


Figura 4.2: ffplay no cliente Portátil 3

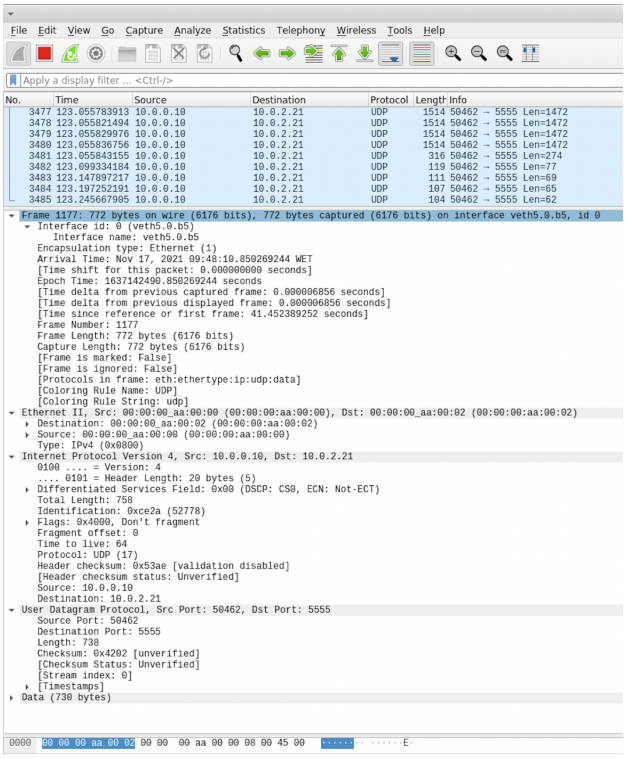


Figura 4.3: Captação do tráfego no wireshark em unicast

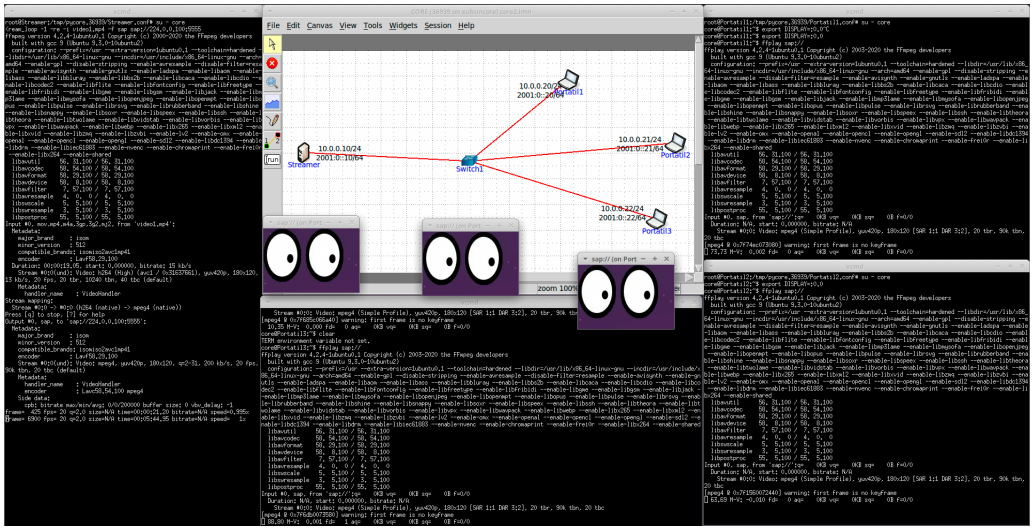


Figura 4.4: Topologia multicast com o streaming a funcionar

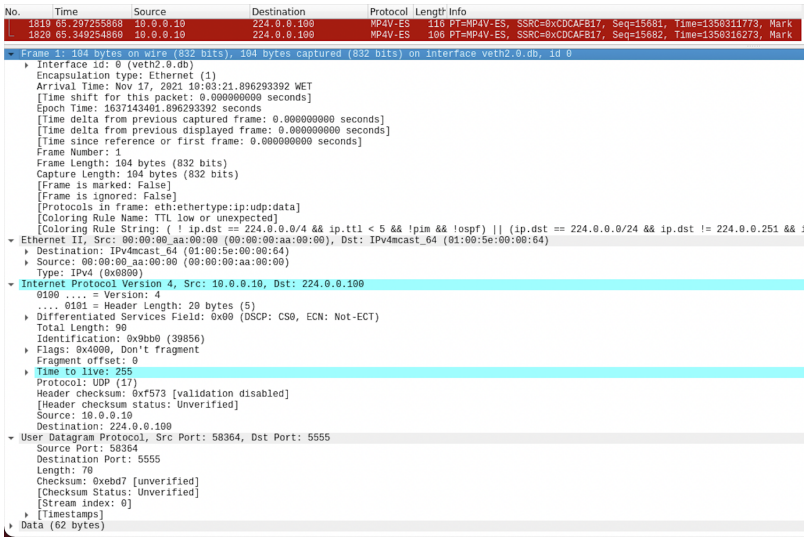


Figura 4.5: Captação do tráfego no wireshark em multicast

**Interfaces**

| Interface | Dropped packets | Capture filter | Link type | Packet size limit |
|---|---|---|---|---|
| veth5.0.b5 | Unknown | none | Ethernet | 262144 bytes |

**Statistics**

| Measurement | Captured | Displayed | Marked |
|---|---|---|---|
| Packets | 6495 | 6495 (100.0%) | — |
| Time span, s | 229.696 | 229.696 | — |
| Average pps | 28.3 | 28.3 | — |
| Average packet size, B | 664 | 664 | — |
| Bytes | 4309772 | 4309772 (100.0%) | 0 |
| Average bytes/s | 18 k | 18 k | — |
| Average bits/s | 150 k | 150 k | — |

Figura 4.6: Estatísticas wireshark em unicast

**Interfaces**

| Interface | Dropped packets | Capture filter | Link type | Packet size limit |
|---|---|---|---|---|
| veth2.0.db | Unknown | none | Ethernet | 262144 bytes |

**Statistics**

| Measurement | Captured | Displayed | Marked |
|---|---|---|---|
| Packets | 3235 | 3235 (100.0%) | — |
| Time span, s | 116.150 | 116.150 | — |
| Average pps | 27.9 | 27.9 | — |
| Average packet size, B | 668 | 668 | — |
| Bytes | 2159621 | 2159621 (100.0%) | 0 |
| Average bytes/s | 18 k | 18 k | — |
| Average bits/s | 148 k | 148 k | — |

Figura 4.7: Estatísticas wireshark em multicast