Conceitualizando:

Ao final da sexta reunião, ficou decidida a implementação do servidor para suportar os procolos previstos (HLS, MPEG-DASH e HTTP) e a análise dos pacotes na rede, tanto rede local quanto externa, visando características de streaming no que tange a velocidade de transmissão, qualidade de vídeo, lagging, entre outros; a fim de que fosse possível determinar o melhor protocolo que será utilizado posteriormente.

Foi pedido também para que fossem analisados artigos referentes aos protocolos e referentes a um “teste de qualidade” de cada protocolo supracitado implementado no nginx.

Nginx e outros protocolos

O que o Nginx suporta?

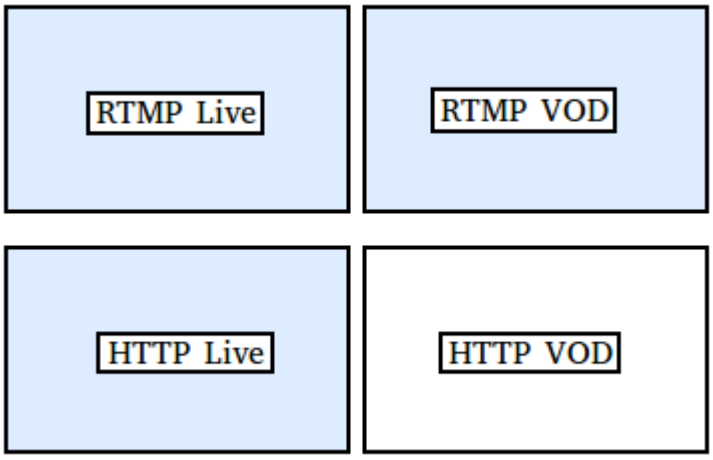

Imagem 1 – Tipos de streaming suportados pelo Nginx.

Em que em HTTP Live e em HTTP VOD[1] são incluídos HLS Live e MPEG-DASH Live.

Um dos pontos centrais para a criação do servidor Nginx é a utilização do módulo RTMP, disponível também na versão free do programa, juntamente com a criação das configurações do servidor, que consiste na implementação dos três diferentes tipos de streaming: RTMP Live, RTMP VOD e HTTP Live, todos marcados na coloração azul claro na figura 1.

Como Payton Sherry [2] considera em seu artigo sobre as decisões e demandas para fazer para a utilização futura do servidor:

1) When to use Live Streaming?
If you have breaking concerns news or live events, you’re probably going to want to live stream it. If you are interested in conducting meetings and collaborating on projects, services like Ready Talk are an excellent option.

2) When to use Video On-Demand?
Corporate training is a good example of when video on-demand might be a better choice. You want all

1) VOD – Video on Demand

of your new employees to be able to access this video, even if it’s not at the same time as everyone else. That’s where Viddler can help. With easy integration and security, Viddler offers high quality video on demand for all types of enterprise solutions.

Definição dada pelo site Stream Video Provider [3]:

Live streaming means taking the video from live sources such as webcams, camcorders, dvd players etc. and broadcasting it live over the Internet to your audience/viewers. If the viewer misses the broadcast the only way to see it is if the publisher provides on-demand streaming.

On-demand streaming means that the publisher needs to upload a pre-recorded video (for example recorded taken during a live event) and then stream it to the viewers who can watch the video 24/7 over and over again.

Criação e configuração do servidor Nginx para suporte aos protocolos HLS e MPEG-DASH

Para criar o streaming que suporte o protocolo HLS, foi necessário primeiramente fazer as alterações nas configurações do servidor, através do arquivo nginx.conf, localizado no diretório /usr/local/nginx/conf.

Inicialmente o servidor não estava suportando o streaming, por falha na configuração do arquivo *nginx.conf*. Assim, foi necessário fazer uma pesquisa extensiva sobre como o Nginx funciona, e como fazer para configurar o live stream para HLS e MPEG-DASH.

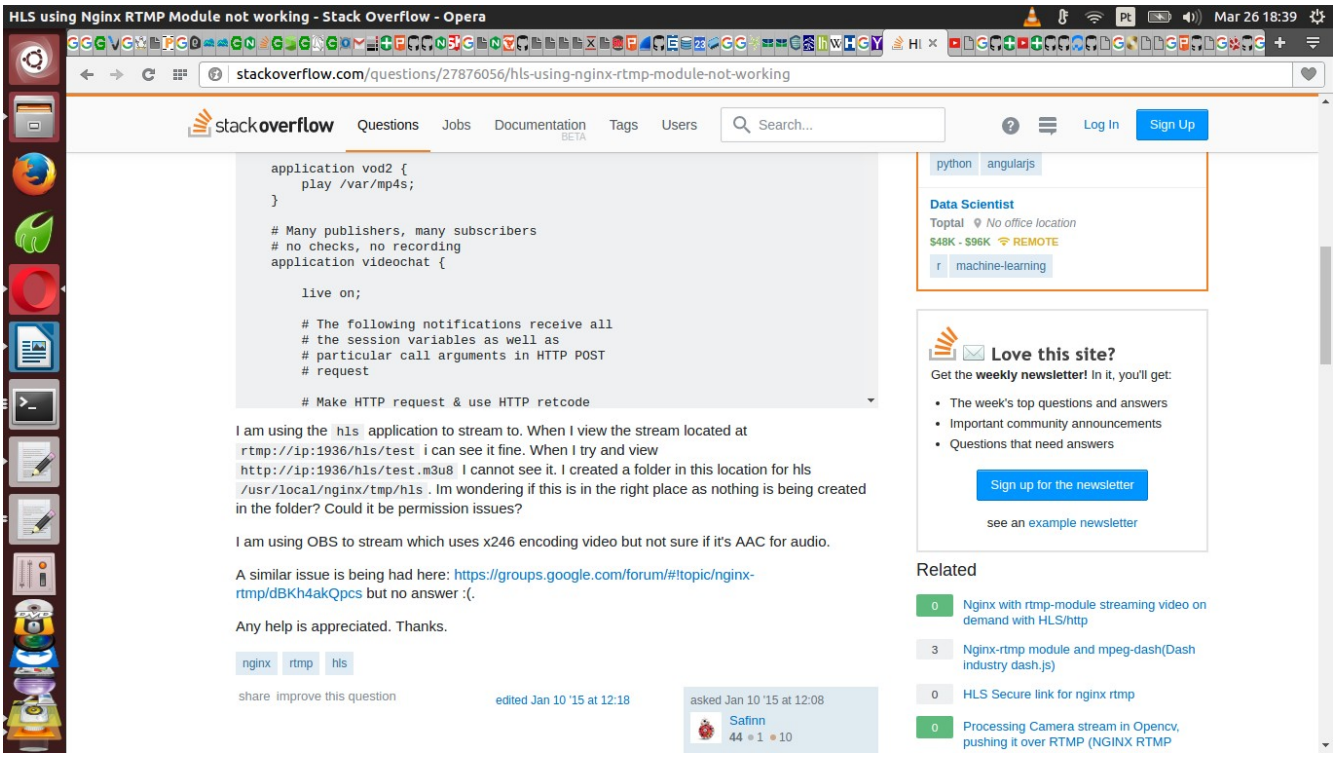

Imagem 2 – Abas de pesquisa no navegador Opera.

Nos testes anteriores foram utilizadas streams de RTMP em VOD, mas, a versão free do software Nginx não possui a aplicação VOD, portanto foi necessário implementar a função Live para

os protocolos requeridos.

Para isso foram necessárias colocar as seguintes configurações, na área das especificações do servidor RTMP, no arquivo nginx.conf, de acordo com [1]:

Para o stream de HLS:

```
application hls {
      live on;
      hls on;
      hls_path /tmp/hls;
}
```

Para o stream de MPEG-DASH:

```
application dash {
      live on;
      dash on;
      dash_path /tmp/dash;
}
```

Em que, ambos para HLS e MPEG-DASH, são definidos parâmetros para a caracterização de cada protocolo e seu tipo de stream. Em HLS foi necessário dizer que ele é Live e dizer que ele é destinado ao protocolo hls, através do comandos, em ordem consecutiva:

```
live on;
hls on;
```

A última linha da aplicação é a dash_path, que diz onde será armazenado o arquivo particionado e a playlist da stream [4].

Para o MPEG-DASH foram definidas basicamente as mesmas configurações, mudando apenas os parâmetros de hls para dash.

Após fazer a alteração no servidor RTMP foi necessário configurar para agora rodar no HTTP com as seguintes configurações:

Para o servidor HLS:

```
location /hls {
      types {
            application/vnd.apple.mpegurl m3u8;
            video/mp2t ts;
      }
      root /tmp;

      add_header Cache-Control no-cache;
}
```

Em que foram definidas algumas características, como o tipo MIME, que de acordo com [5]: “The MIME type is the mechanism to tell the client the variety of document transmitted: the extension of a file name has no meaning on the web. It is, therefore, important that the server is correctly set up,

so that the correct MIME type is transmitted with each document. Browsers often use the MIME-type to determine what default action to do when a resource is fetched", para o stream HTTP como sendo m3u8, conforme especificado no arquivo *mime.types*, armazenado no diretório */usr/local/nginx/conf/*, que consiste em definir a compressão do arquivo particionado. Essa definição serve para carregar um cache do arquivo que está sendo realizado o stream, como é explicado em 19:46 até 24:03 em [1]:

Vídeo 1 – Explicação dos formatos m3u8 e MP4Box.

Já o outro tipo definido para a stream, o ts, é utilizado para armazenar as múltiplas partes do formato do conteiner de dados [6], como pode ser visto um exemplo no vídeo 1. Por sua vez a última linha da definição do hls no HTTP é *add_header Cache-Control no-cache;* em que, segundo a Oracle [7], o parâmetro *no-cache* "Disables caching completely and overrides Web Service editor settings", de acordo com o diagrama abaixo:

public	Allows any cached content to be shared across users with identical sets of preferences using the same portal server. This value should be used whenever possible.
private	Tells the portal server not to share cached content. The User ID is added to the cache key so that a separate copy is retained in the cache for each individual user. This value should only be used to protect sensitive information, for example, an e-mail inbox portlet. (User settings can also make public content effectively private.)
max-age= [seconds]	Specifies the maximum amount of time that an object is considered fresh. Similar to the Expires header, this directive allows more flexibility. [seconds] is the number of seconds from the time of the request that the object should remain fresh.
must-revalidate	Tells the cache that it must obey any freshness information it receives about an object. HTTP allows caches to take liberties with the freshness of objects; specifying this header tells the cache to strictly follow your rules.
no-cache	Disables caching completely and overrides Web Service editor settings. Neither the client nor the Portal Server responds to subsequent requests with a cached version.

Imagem 3 – Definições do parâmetro no-cache [7].

Para o servidor MPEG-DASH:

```
location /dash {
      root /tmp;
      add_header Cache-Control no-cache;
}
```

Suas definições são as mesmas definiçãos para o HLS.

Caracterização dos meios de como realizar a stream desejada:

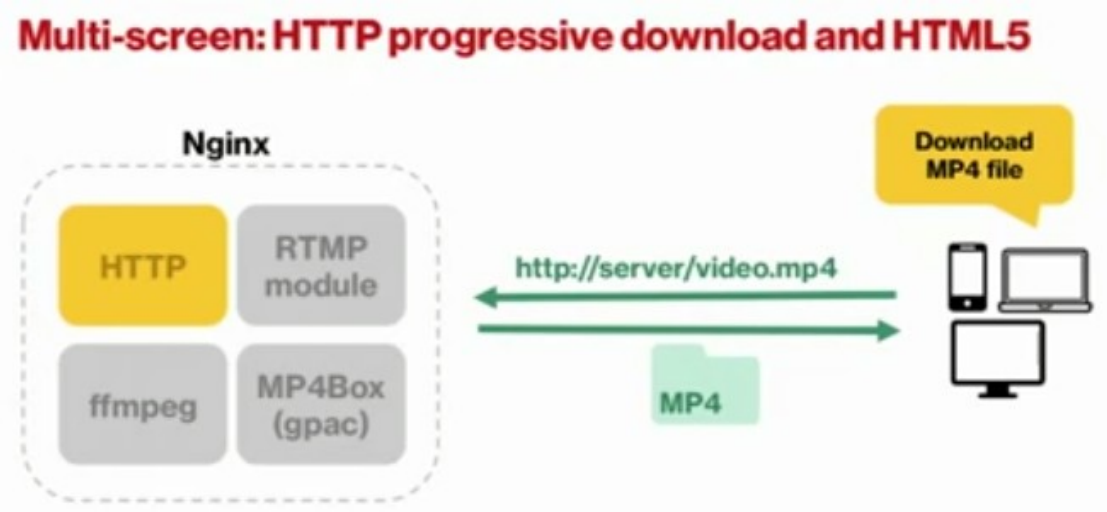

Imagem 4 – Requisição de arquivo, VOD, via HTTP [1].

O cliente faz uma requisição para o servidor do arquivo a ser enviado. Assim, o servidor HTTP retorna o arquivo para o cliente, para que faça download.

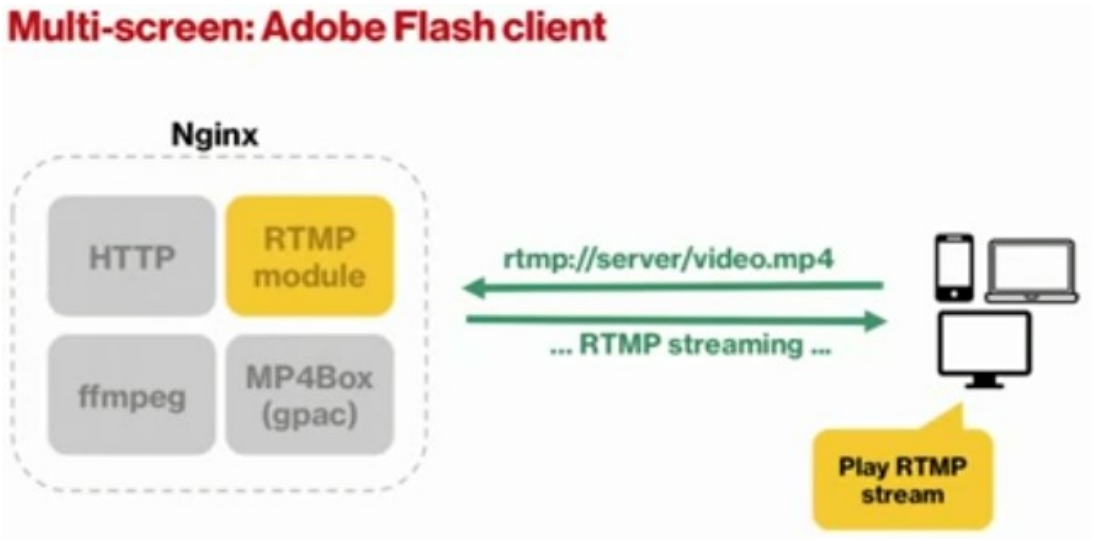

Imagem 5 – Requisição de stream via RTMP [1].

O cliente faz uma requisição via RTMP para o servidor, em que ele, por sua vez, retorna a stream desejada para o cliente.

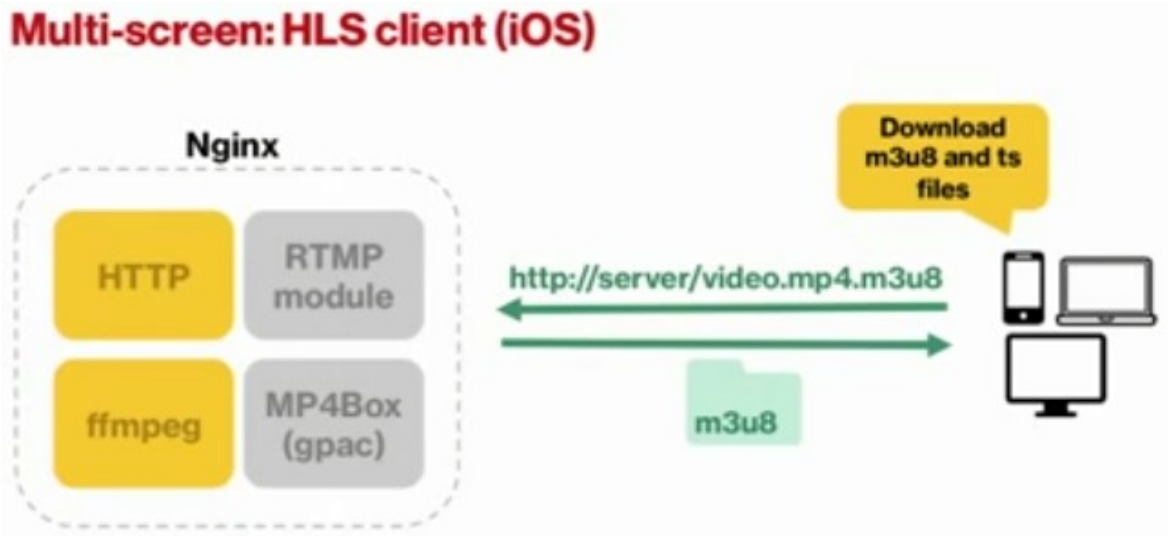

Imagem 6 – Requisição de stream sob o protocolo HLS [1].

O cliente faz uma requisição para o servidor HTTP de um arquivo .m3u8, caracterizando a stream sob o protocolo HLS.

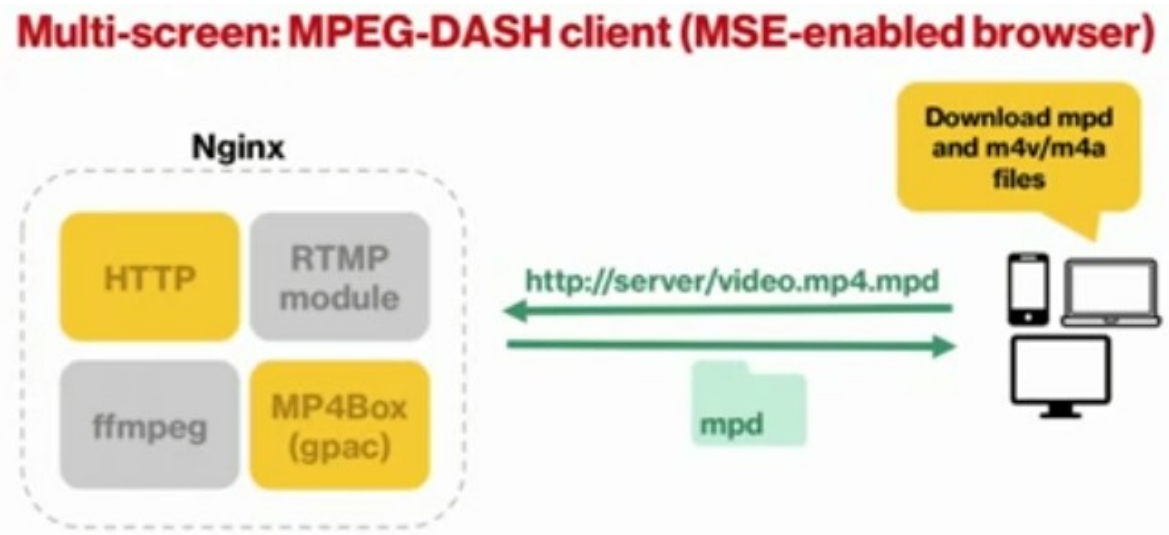

Imagem 7 – Requisição de stream sob o protocolo MPEG-DASH [1].

O cliente faz uma requisição para o servidor HTTP de um arquivo .mpd, caracterizando a stream sob o protocolo MPEG-DASH.

Realizando a stream do HLS

Para que fosse possível realizar a stream devidamente correta e funcional foi necessária fazer a utilização do FFmpeg, o uma ferramenta que permite a reprodução, edição e o stream de vídeos e áudios, ou, como o próprio site define “A complete, cross-platform solution to record, convert and stream audio and video” [12].

Imagem 8 – FFmpeg

Deste modo, foi necessária a utilização da seguinte linha de comando, de acordo com a definição aproximada em [13]:

```
ffmpeg -re -i
/home/nothereboy/Documents/Nginx_Server_Streamming_Files/hls/live/Serenity.mp4
-vcodec libx264 -g 30 -acodec aac -strict -2 -f flv rtmp://172.31.184.84/hls/stream
```

Em que os seguintes parâmetros podem ser definidos como:

- re – consome a taxa de bits de acordo com a nativa da mídia, e não o mais rápido possível.
- i – seleciona o dispositivo físico para ser capturado, seleciona o vídeo para ser realizado o stream.
- vcodec – especifica o codificador de vídeo para a saída. Foi utilizado o x264 H.264/MPEG-4 [14].
- g - define uma coleção de imagens sucessivas dentro de um fluxo de vídeo codificado [15].
- acodec – especifica o codificador de áudio para a saída. Foi utilizado o AAC [14].
- strict -2 – permite utilizar o codec AAC (Essa definição é para o strict, não consegui achar o que era -strict -2).
- f – especifica o formato de vídeo para a saída.
- rtmp://172.31.184.84/hls/stream – é o endereço da stream. O campo stream corresponde ao nome da stream, podendo, assim, realizar várias streams.

Por prova de conceito, foi visualizada a pasta /tmp/hls/ para verificar se os arquivos da stream estavam sendo gerados realmente.

```
nothereboy@nothereboy:~$ sudo ls /tmp/hls/
stream-0.ts  stream-2.ts  stream-4.ts  stream.m3u8
stream-1.ts  stream-3.ts  stream-5.ts
```

Imagem 9 – Arquivos gerados pelo FFmpeg.

Resultados da stream do HLS

A stream foi realizada com sucesso com a utilização do FFmpeg, serultando nos seguintes metadados:

```
major_brand     : isom
minor_version   : 1
compatible_brands: isomavc1
date            : 2005
genre           : Trailer
artist          : Universal Pictures
title           : Serenity - HD DVD Trailer
encoder         : Lavf56.4.101
Stream #0:0(und): Video: h264 (libx264) ([7][0][0][0] / 0x0007), yuv420p, 1280x720,
q=-1--1, 23.98 fps, 1k tbn, 23.98 tbc (default)
```

```
creation_time   : 2007-05-30 05:20:31
handler_name    : GPAC ISO Video Handler
encoder         : Lavc56.1.100 libx264
Stream #0:1(und): Audio: aac ([10][0][0][0] / 0x000A), 48000 Hz, stereo, fltp, 128
kb/s (default)
creation_time   : 2007-05-30 05:20:34
handler_name    : GPAC ISO Audio Handler
encoder         : Lavc56.1.100 aac

Stream mapping:
   Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
   Stream #0:1 -> #0:1 (aac (native) -> aac (native))

video:11954kB audio:844kB subtitle:0kB other streams:0kB global headers:0kB muxing
overhead: 0.528107%
[libx264 @ 0x1136580] frame I:35    Avg QP:17.22  size: 54004
[libx264 @ 0x1136580] frame P:336   Avg QP:22.46  size: 10884
[libx264 @ 0x1136580] frame B:325   Avg QP:26.29  size:  1739
[libx264 @ 0x1136580] consecutive B-frames: 32.8% 12.9%  6.0% 48.3%
[libx264 @ 0x1136580] mb I  I16..4: 22.6% 64.4% 13.0%
[libx264 @ 0x1136580] mb P  I16..4:  6.7% 16.2%  0.9%  P16..4: 29.2%  6.6%  3.1%
0.0%  0.0%    skip:37.4%
[libx264 @ 0x1136580] mb B  I16..4:  0.1%  0.2%  0.0%  B16..8: 23.9%  0.9%  0.1%
direct: 0.4%  skip:74.2%  L0:42.8% L1:53.2% BI: 4.0%
[libx264 @ 0x1136580] 8x8 transform intra:66.9% inter:80.1%
[libx264 @ 0x1136580] coded y,uvDC,uvAC intra: 35.6% 39.7% 15.2% inter: 8.2% 9.0%
0.5%
[libx264 @ 0x1136580] i16 v,h,dc,p: 41% 28% 12% 19%
[libx264 @ 0x1136580] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 31% 19% 27%  3%  4%  4%  4%
4%  4%
[libx264 @ 0x1136580] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 29% 24% 14%  5%  6%  6%  6%
5%  4%
[libx264 @ 0x1136580] i8c dc,h,v,p: 61% 17% 17%  4%
[libx264 @ 0x1136580] Weighted P-Frames: Y:5.7% UV:2.7%
[libx264 @ 0x1136580] ref P L0: 75.2% 12.3%  9.0%  3.5%  0.0%
[libx264 @ 0x1136580] ref B L0: 90.7%  8.3%  1.1%
[libx264 @ 0x1136580] ref B L1: 95.4%  4.6%
[libx264 @ 0x1136580] kb/s:1684.55
```

Imagem 10 – Stream sendo carregada.

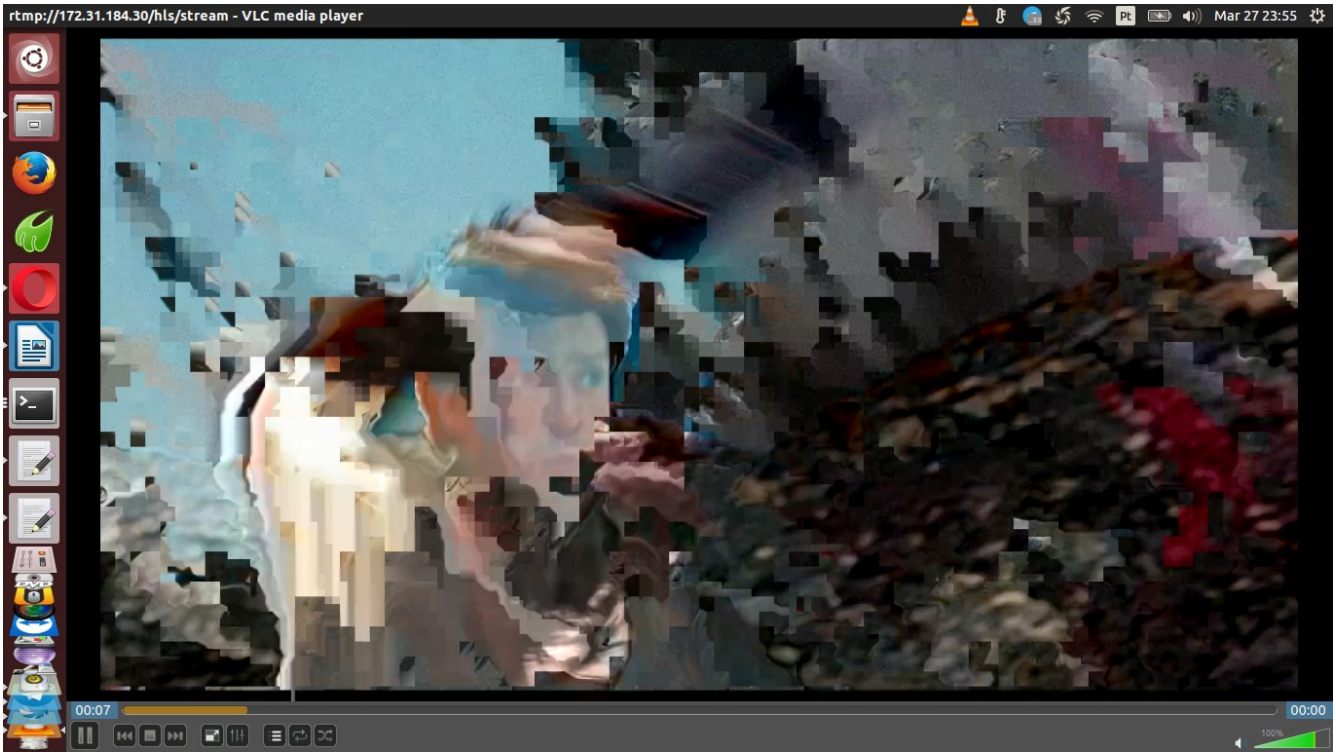

Imagem 11 – Stream sendo carregada.

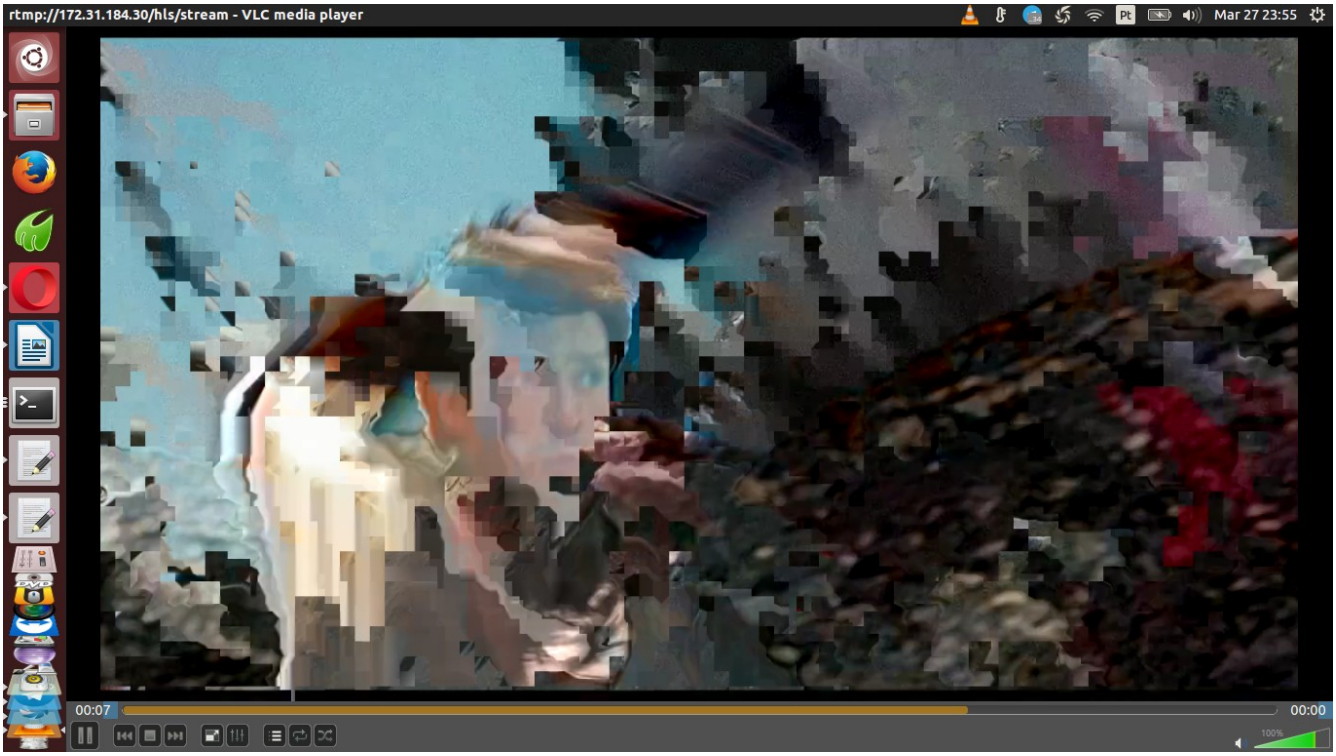

Imagem 12 – Stream sendo carregada.

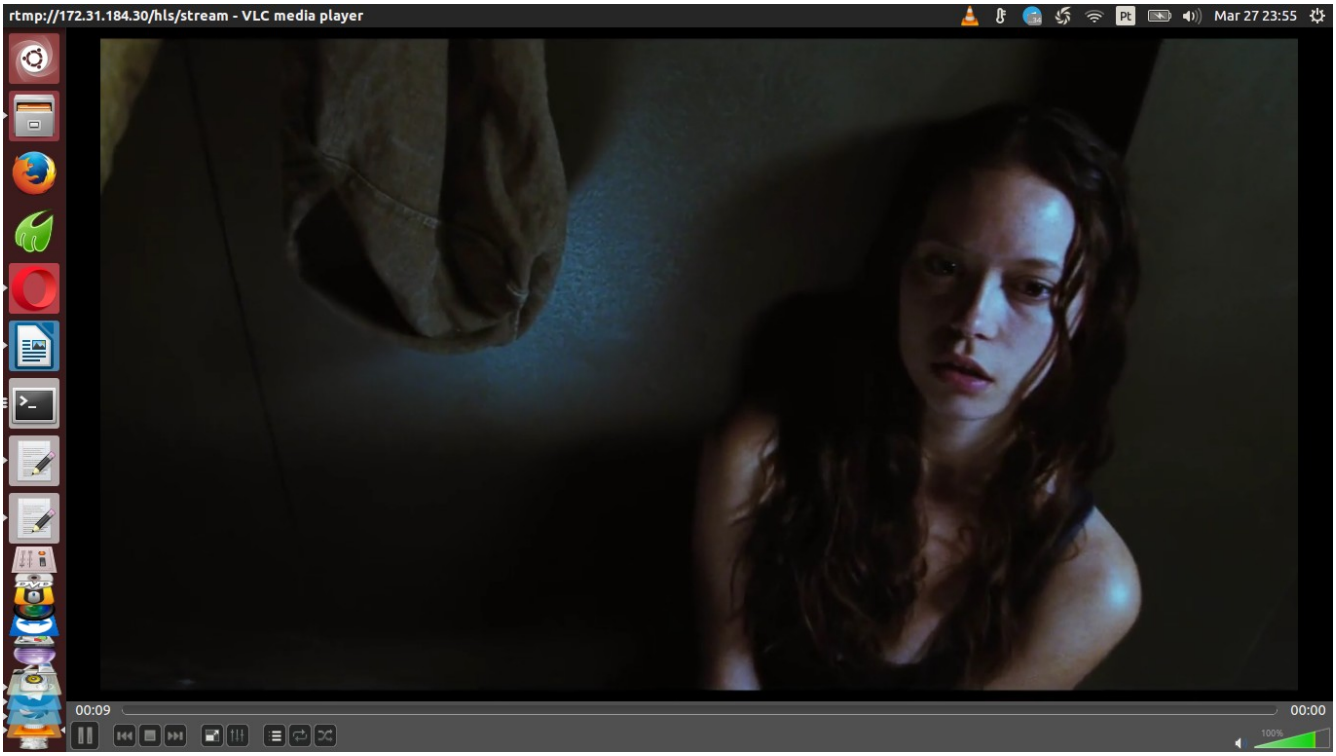

Imagem 13 – Stream carregada.

Análise dos resultados da stream utilizando o protocolo HLS

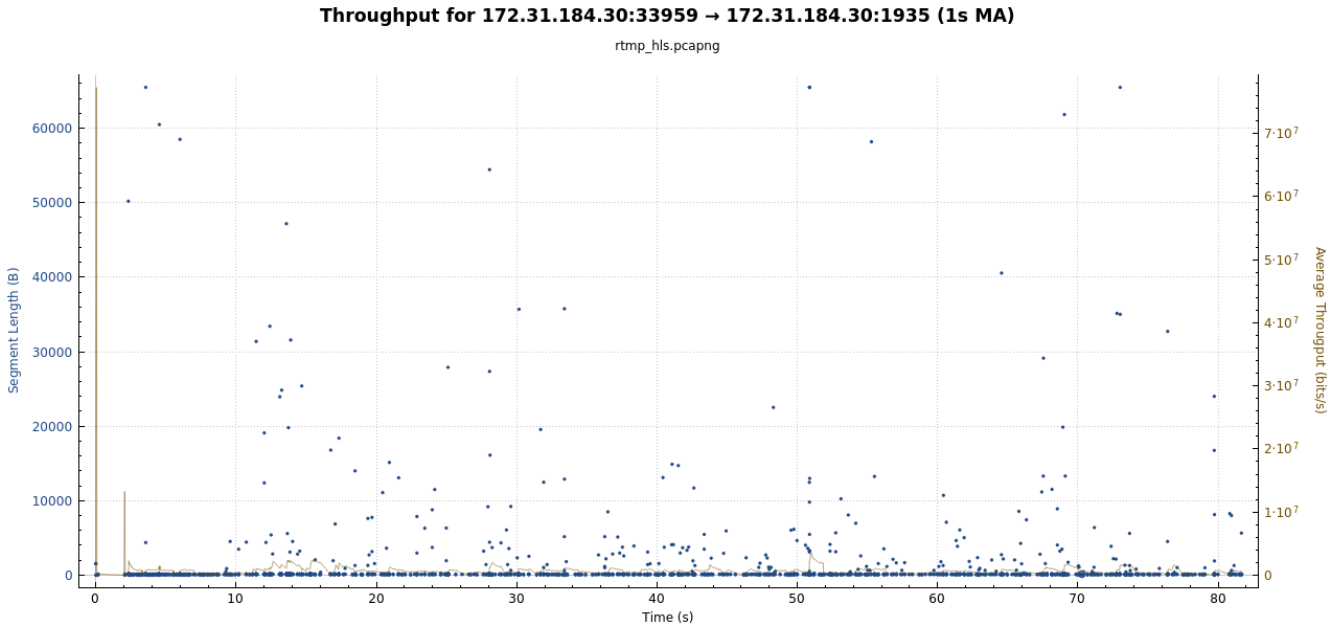

Imagem 14: Vazão do streaming do servidor para o cliente.

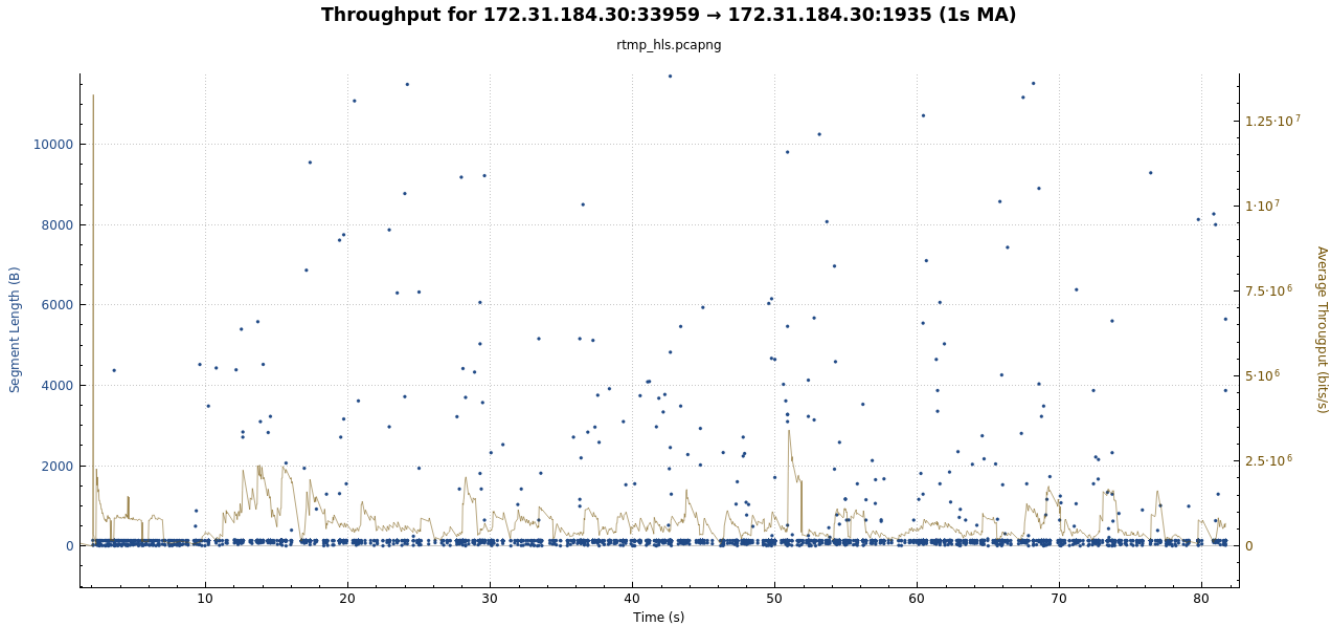

Imagem 15: Zoom da vazão do streaming do servidor para o cliente.

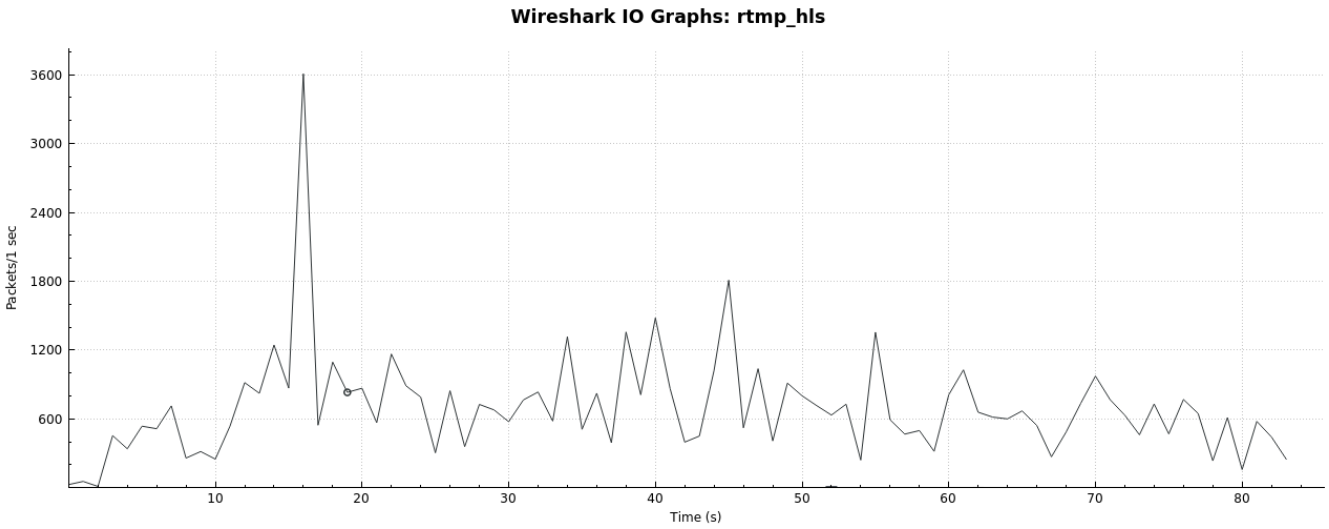

Imagem 16 – Gráfico de I/O.

Time	172.31.184.30	Comment
1.527792086	1935 → 33959	RTMP: Handshake C0+C1
1.527929567	33959 → 1935	RTMP: Handshake S0+S1+S2
1.528003786	1935 → 33959	RTMP: Handshake C2
1.528102936	1935 → 33959	RTMP: connect('hls')
1.528197360	33959 → 1935	RTMP: Window Acknowledgement Size 5000000
1.565977022	33959 → 1935	RTMP: Set Peer Bandwidth 5000000,Dynamic|Set...
1.566266689	1935 → 33959	RTMP: releaseStream('stream')
1.566293990	1935 → 33959	RTMP: FCPublish('stream')
1.566317519	1935 → 33959	RTMP: createStream()
1.566453908	33959 → 1935	RTMP: _result()
1.566596707	1935 → 33959	RTMP: publish('stream')
1.697308135	1935 → 33959	RTMP: @setDataFrame()
1.697347705	1935 → 33959	RTMP: Video Data
1.697385220	1935 → 33959	RTMP: Audio Data
3.586152686	1935 → 33959	RTMP: Video Data
3.596042215	1935 → 33959	RTMP: Audio Data
3.596154660	1935 → 33959	RTMP: Video Data
3.610693572	1935 → 33959	RTMP: Audio Data
3.610808188	1935 → 33959	RTMP: Audio Data
3.610904255	1935 → 33959	RTMP: Video Data
3.841138521	1935 → 33959	RTMP: Audio Data
3.841374150	1935 → 33959	RTMP: Audio Data
3.842169388	1935 → 33959	RTMP: Audio Data
3.842220213	1935 → 33959	RTMP: Unknown (0x0)

Imagem 17 – Flow Graph. Intervalo de tempo de 1.52s a 3.84s.

Imagem 18 – Flow Graph. Intervalo de tempo de 7.07s a 7.30s.

Statistics

Measurement	Captured	Displayed	Marked
Packets	58470	5867 (10.0%)	N/A
Time span, s	83.357	81.653	N/A
Average pps	701.4	71.9	N/A
Average packet size, B	260.5	1146.5	N/A
Bytes	15257771	6729034 (44.1%)	0
Average bytes/s	183 k	82 k	N/A
Average bits/s	1,464 k	659 k	N/A

Imagem 19 – Resumo da transmissão de pacotes.

Pacotes e portas por IP:

Topic / Item	Count	Rate (ms)	Percent	Burst rate	Burst start
172.31.184.30	56308	0.6755	96.31%	24.4700	16.790
TCP	56308	0.6755	100.00%	24.4700	16.790
51680	2	0.0000	0.00%	0.0100	17.449
50189	4	0.0000	0.01%	0.0100	7.393
33960	1755	0.0211	3.12%	0.2000	18.844
33959	24891	0.2986	44.21%	12.1800	16.790
1935	29656	0.3558	52.67%	12.2300	16.790

Realizando a stream do MPEG-DASH

Para executar a stream do protocolo MPEG-DASH foi necessário utilizar a ferramenta Ffmpeg, utilizando a seguinte linha de comando:

```
ffmpeg -re -i /home/nothereboy/Documents/Nginx_Server_Streamming_Files/hls/live/Serenity.mp4 -vcodec libx264 -g 30 -acodec aac -strict -2 -f flv rtmp://172.31.184.30/dash/stream
```

Por prova de conceito, foi visualizada a pasta /tmp/dash/ para verificar se os arquivos da stream estavam sendo gerados realmente.

```
nothereboy@nothereboy:~$ sudo ls /tmp/dash/
stream-16141.m4a  stream-27194.m4a  stream-37621.m4a  stream.mpd
stream-16141.m4v  stream-27194.m4v  stream-37621.m4v  stream-raw.m4a
stream-21188.m4a  stream-32199.m4a  stream-init.m4a   stream-raw.m4v
```

Imagem 20 – Arquivos gerados pelo FFmpeg.

Resultados da stream do MPEG-DASH

Foram obtidos os seguintes metadados após a execução do FFmpeg:

```
major_brand     : isom
minor_version   : 1
compatible_brands: isomavc1
date            : 2005
genre           : Trailer
artist          : Universal Pictures
title           : Serenity - HD DVD Trailer
encoder         : Lavf56.4.101
Stream #0:0(und): Video: h264 (libx264) ([7][0][0][0] / 0x0007), yuv420p, 1280x720,
q=-1--1, 23.98 fps, 1k tbn, 23.98 tbc (default)
creation_time   : 2007-05-30 05:20:31
handler_name    : GPAC ISO Video Handler
encoder         : Lavc56.1.100 libx264
Stream #0:1(und): Audio: aac ([10][0][0][0] / 0x000A), 48000 Hz, stereo, fltp, 128
kb/s (default)
creation_time   : 2007-05-30 05:20:34
handler_name    : GPAC ISO Audio Handler
encoder         : Lavc56.1.100 aac
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:1 -> #0:1 (aac (native) -> aac (native))

video:4171kB audio:326kB subtitle:0kB other streams:0kB global headers:0kB muxing
overhead: 0.588748%
[libx264 @ 0x2480580] frame I:25    Avg QP:16.77  size: 53280
[libx264 @ 0x2480580] frame P:231   Avg QP:22.65  size: 10809
[libx264 @ 0x2480580] frame B:246   Avg QP:26.75  size:  1797
[libx264 @ 0x2480580] consecutive B-frames: 28.5% 15.9%  7.8% 47.8%
[libx264 @ 0x2480580] mb I  I16..4: 24.3% 62.9% 12.9%
[libx264 @ 0x2480580] mb P  I16..4:  5.8% 15.2%  0.8%  P16..4: 28.9%  6.6%  3.1%
0.0%  0.0%    skip:39.7%
[libx264 @ 0x2480580] mb B  I16..4:  0.2%  0.3%  0.0%  B16..8: 21.2%  1.0%  0.2%
direct: 0.5%  skip:76.6%  L0:41.3% L1:53.8% BI: 5.0%
[libx264 @ 0x2480580] 8x8 transform intra:67.2% inter:80.5%
[libx264 @ 0x2480580] coded y,uvDC,uvAC intra: 35.5% 35.7% 13.0% inter: 8.1% 8.3%
0.4%
[libx264 @ 0x2480580] i16 v,h,dc,p: 43% 30% 11% 17%
[libx264 @ 0x2480580] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 31% 20% 26%  3%  4%  4%  4%
4%  4%
[libx264 @ 0x2480580] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 29% 25% 13%  5%  6%  6%  6%
```

```
5%  5%
[libx264 @ 0x2480580] i8c dc,h,v,p: 64% 17% 16%  3%
[libx264 @ 0x2480580] Weighted P-Frames: Y:7.8% UV:3.9%
[libx264 @ 0x2480580] ref P L0: 75.4% 12.6%  8.9%  3.1%  0.0%
[libx264 @ 0x2480580] ref B L0: 90.6%  8.4%  1.0%
[libx264 @ 0x2480580] ref B L1: 95.6%  4.4%
[libx264 @ 0x2480580] kb/s:1631.81
```

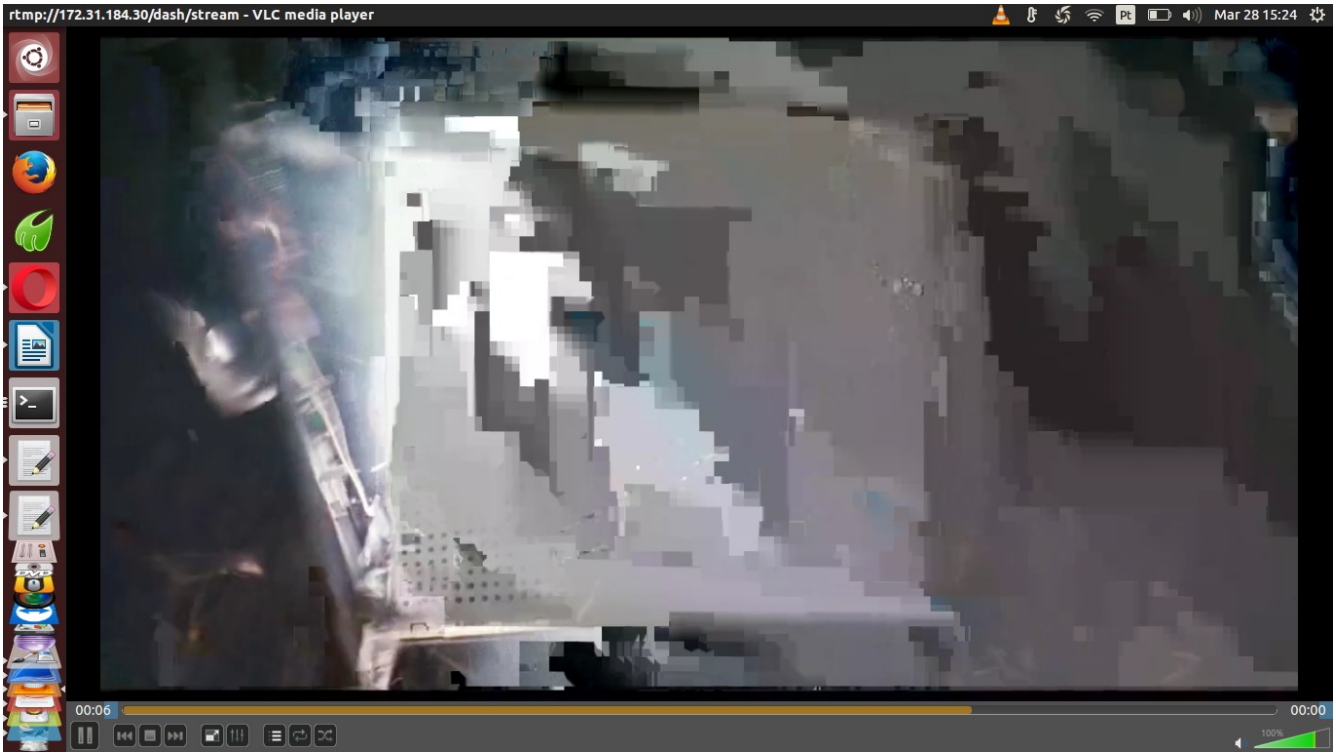

Imagem 21 – Stream sendo carregada.

Imagem 22 – Stream carregada.

Análise dos resultados da stream utilizando o protocolo MPEG-DASH

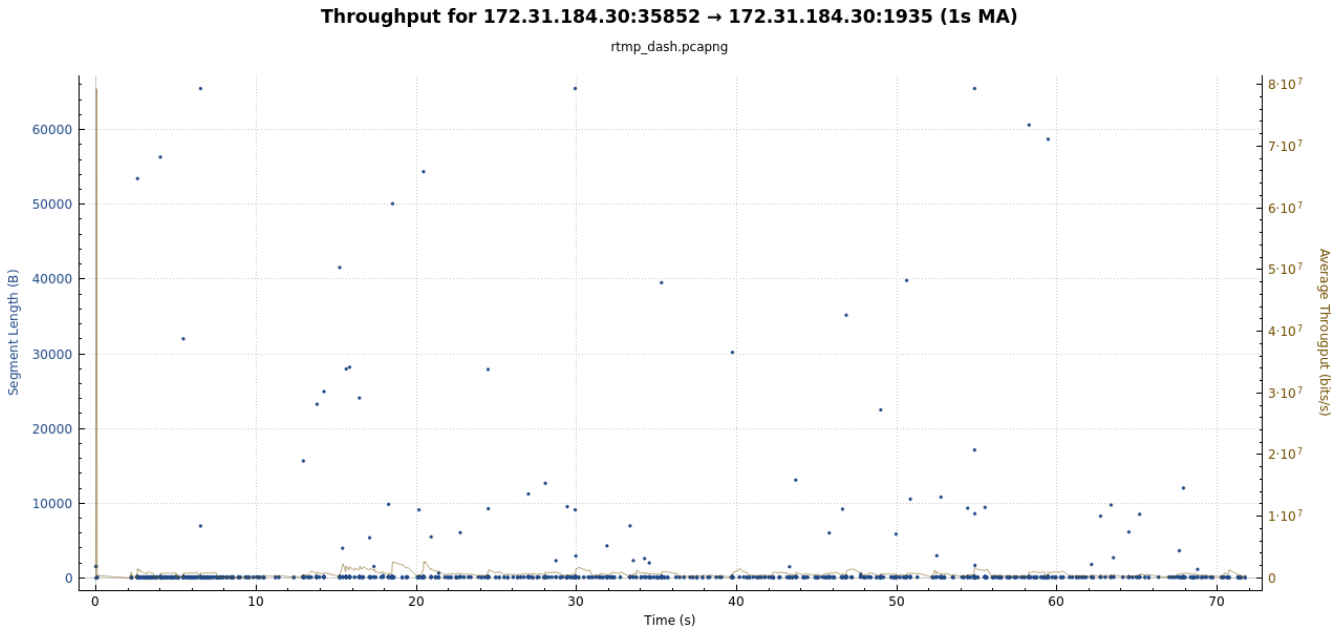

Imagem 23: Vazão do streaming do servidor para o cliente.

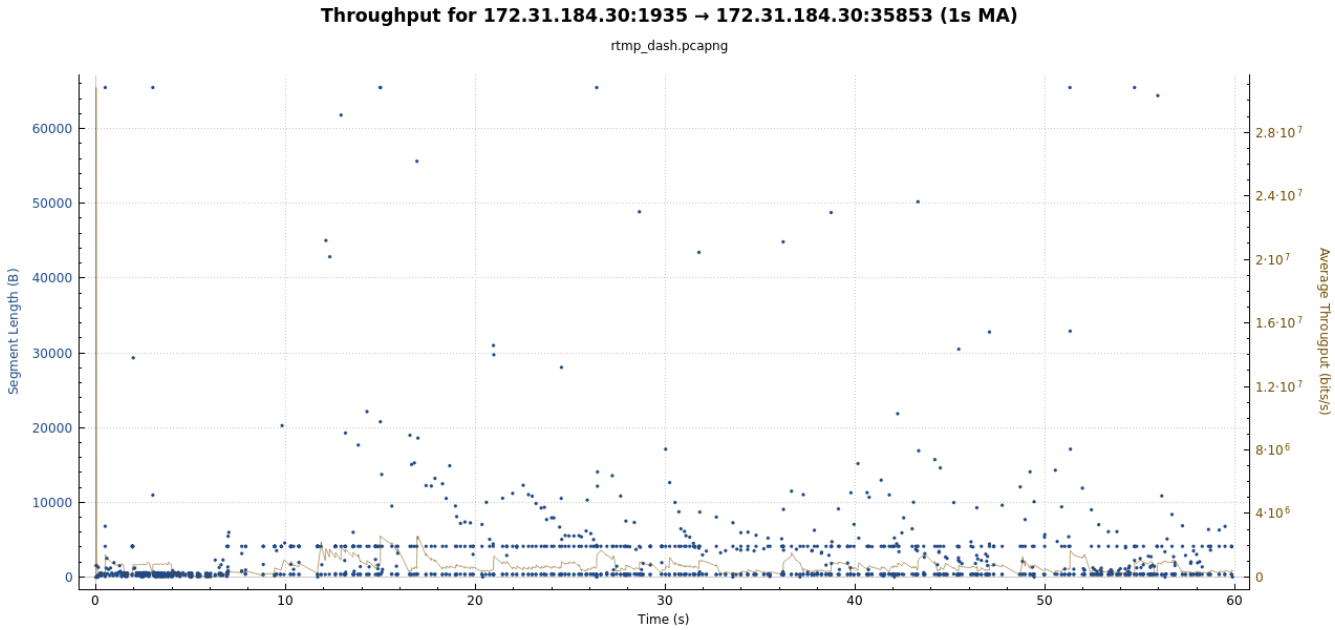

Imagem 24 – Zoom da vazão do streaming do servidor para o cliente.

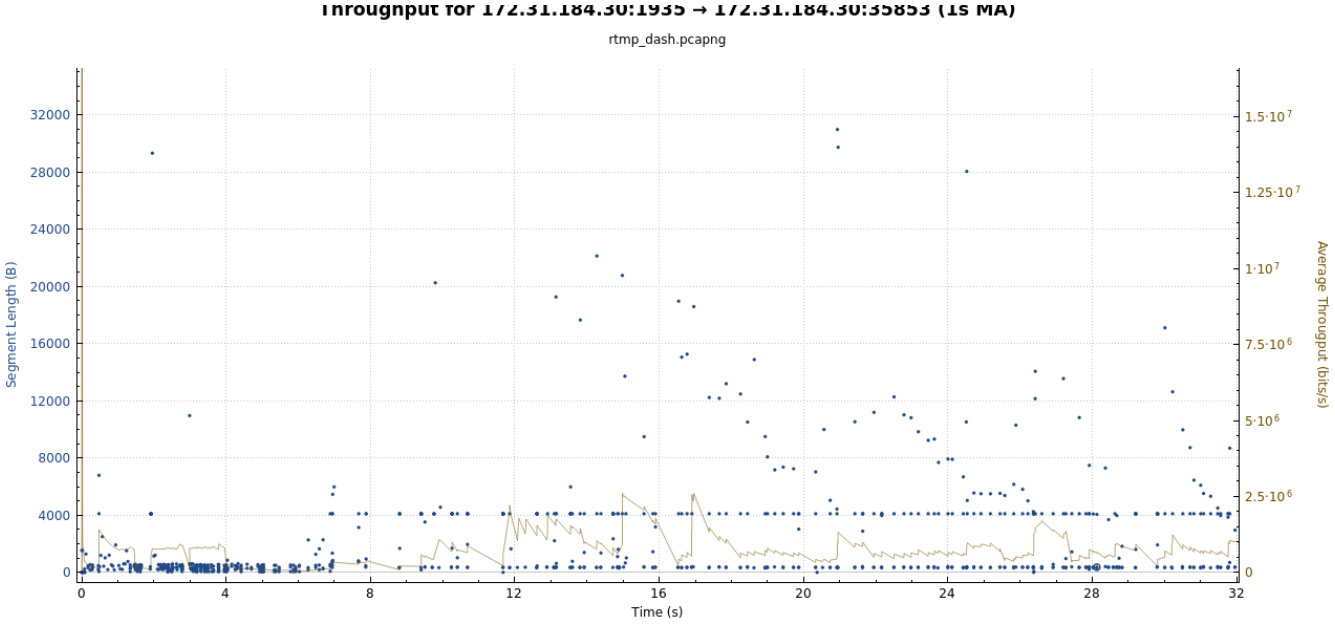

Imagem 25 – Maior zoom da vazão do streaming do servidor para o cliente.

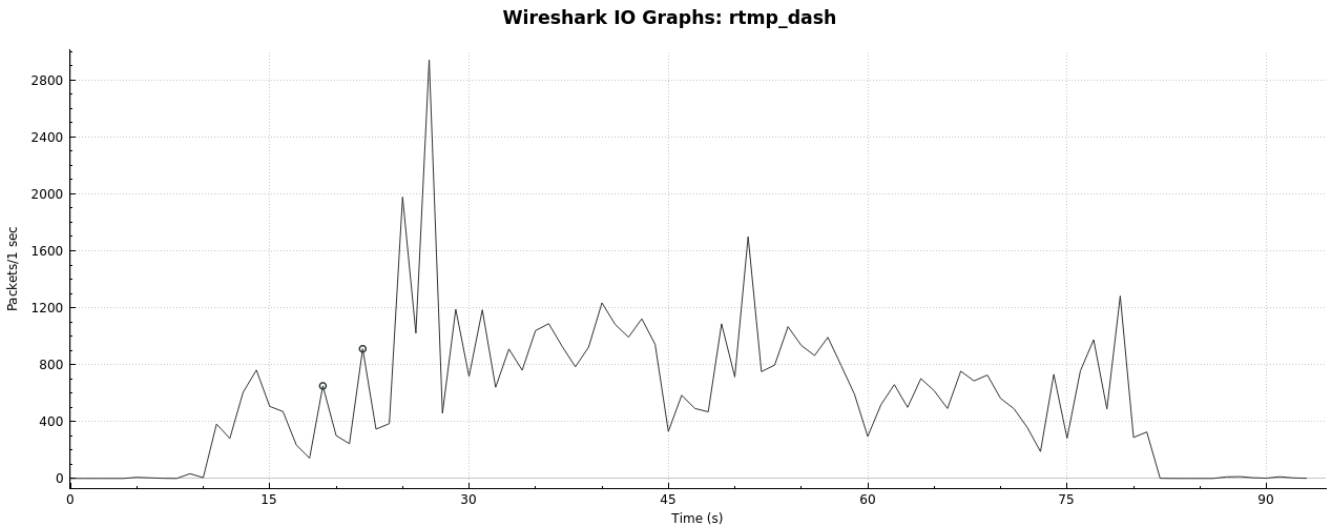

Imagem 26 – Gráfico de I/O.

Imagem 27 – Flow Graph para a inicilização do servidor. Intervalo de tempo de 9.31s a 11.89s.

Imagem 28 – Flow Graph para a inicilização da stream. Intervalo de tempo de 12.85s a 13.03s.

Measurement	Captured	Displayed	Marked
Packets	53157	4929 (9.3%)	N/A
Time span, s	93.189	71.746	N/A
Average pps	570.4	68.7	N/A
Average packet size, B	233.5	1031.5	N/A
Bytes	12432345	5086582 (40.9%)	0
Average bytes/s	133 k	70 k	N/A
Average bits/s	1,067 k	567 k	N/A

Imagem 30 – Resumo da transmissão de pacotes.

Topic / Item	Count	Rate (ms)	Percent	Burst rate	Burst start
172.31.184.30	52912	0.5678	99.54%	16.2600	27.752
TCP	52912	0.5678	100.00%	16.2600	27.752
35853	1393	0.0149	2.63%	0.1600	15.166
35852	23106	0.2479	43.67%	7.8100	27.752
1935	28413	0.3049	53.70%	8.3900	27.752

Comparação entre protocolos

Foi, então, realizada um request simultâneo da stream de cada um dos protocolos estudados: RTMP, HLS e MPEG-DASH.
Foram utilizados os programas VLC, para a visualização do stream, e o Wireshark para a análise da stream.

Resultado da transmissão RTMP:

Tentativas

Foi realizado, por enquanto sem sucesso (aparecia o seguinte erro quando tentava abrir o stream no navegador: “error loading player no playable sources found”), a visualização do streaming através de um navegador, mas as configurações não estão permitindo ainda a exibição da stream, com o servidor rodando. Para que seja possível realizar tal operação, até onde eu pesquisei, é necessária a utilização de um plugin java, chamado jwplayer.js e alguns módulos que trabalharão em conjunto com o jwplayer, sendo eles o jwplayer.flash.swf e o jwplayer.html5.js.

Para que seja possível visualizar o plugin é necessário que seja chamado através de um html. Sendo assim, o arquivo *streaming.html* será configurado como se segue:

```
<!DOCTYPE HTML>
<html>
	<head>
		<meta http-equiv="Content-type" content="text/html; charset=utf-8">
		<title> Live 1 - 1080p</title>

		<script type="text/javascript" src="jwplayer.js"></script>
	</head>

	<body>
		<script type="text/javascript" src="/jwplayer/jwplayer.js"></script>
		<div id="myElement">Loading the player...</div>

		<script type="text/javascript">
			jwplayer("myElement").setup({
			file: "rtmp://192.168.0.10:1935/hls/stream/flv:1080"
			});
		</script>
	</body>
</html>
```

A escrita do código html foi feito de acordo com duas referências: *How To Stream Videos With Nginx and JWPlayer on CentOS 6* [8] e *Nginx-RTMP - 2a - Configure and do your first test Stream* [9]. Já a configuração do jwplayer pode ser encontrada em [10].

Os arquivos acima citados podem ser encontrados em [11].

Importante

Para o protocolo HLS[13]:

Capture webcam on /dev/video0 and stream it to nginx :

```
ffmpeg -re -f video4linux2 -i /dev/video0 -vcodec libx264 -vprofile baseline
```

```
-acodec aac -strict -2 -f flv rtmp://localhost/show/stream
```

Stream file example-vid.mp4

```
ffmpeg -re -i example-vid.mp4 -vcodec libx264 -vprofile baseline -g 30 -acodec aac
-strict -2 -f flv rtmp://localhost/show/stream
```

Stream another rtmp stream :

```
ffmpeg -i rtmp://example.com/appname/streamname -vcodec libx264 -vprofile baseline
-acodec aac -strict -2 -f flv rtmp://localhost/show/stream
```

Referências

[1] https://www.youtube.com/watch?v=t8ebB9Pxb2s → Streaming HLS and DASH with NGINX | Verizon. Acesso em 25/03.
[2] https://www.viddler.com/blog/cracking-the-code-on-video-live-streaming-vs-video-on-demand/ → Live Streaming vs Video On-demand (VOD). Acesso em 25/03.
[3] http://www.streamingvideoprovider.co.uk/kb/index.php/article/what-is-the-difference-between-live-streaming-and-on-de → What Is The Difference Between Live Streaming And On-Demand Streaming? Acesso em 25/03.
[4] https://github.com/arut/nginx-rtmp-module/wiki/Directives#pull → Directives. Acesso em 25/03.
[5] https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types → MIME types. Acesso em 27/03.
[6] http://whatis.techtarget.com/fileformat/TS-HDTV-sample-file-Transport-Stream-MPEG-2-video-stream → TS File Format . Acesso em 27/03.
[7] https://docs.oracle.com/cd/E13158_01/alui/wci/docs103/devguide/tsk_pagelets_settingcaching_httpcachecontrol.html → Setting HTTP Caching Headers – Cache-Control. Acesso em 27/03.
[8] https://www.digitalocean.com/community/tutorials/how-to-stream-videos-with-nginx-and-jwplayer-on-centos-6 → How To Stream Videos With Nginx and JWPlayer on CentOS 6. Acesso em 25/03.
[9] https://www.youtube.com/watch?v=eg69VD2VUo8 → Nginx-RTMP - 2a - Configure and do your first test Stream . Acesso em 26/03.
[10] https://github.com/jwplayer/jwplayer → JW Player. Acesso em 26/03.
[11] https://videos.cdn.mozilla.net/uploads/air_mozilla/players/jwplayer6/ → Index of jwplayer6. Acesso em 25/03.
[12] https://ffmpeg.org → FFmpeg. Acesso em 27/03.
[13] https://docs.peer5.com/guides/setting-up-hls-live-streaming-server-using-nginx/ → Setting up HLS live streaming server using NGINX + nginx-rtmp-module on Ubuntu. Acesso em 26/03.
[14] https://www.ffmpeg.org/ffmpeg-codecs.html#Video-Encoders → FFmpeg Codecs Documentation. Acesso em 27/03.
[15] https://ffmpeg.org/pipermail/ffmpeg-user/2015-June/026945.html → ffmpeg -g option. Acesso em 27/03.

[] https://github.com/arut/nginx-rtmp-module/issues/758 → Perfect FFMPEG configuration to publish on RTMP Nginx #758. Acesso em 25/03.
[] https://docs.peer5.com/guides/setting-up-hls-live-streaming-server-using-nginx/ → Setting up HLS live streaming server using NGINX + nginx-rtmp-module on Ubuntu. Acesso em 25/03.

[99] https://serversforhackers.com/nginx-caching → Nginx Caching.
[99] https://licson.net/post/setting-up-adaptive-streaming-with-nginx/ → Setting Up Adaptive Streaming with Nginx.
[99] https://gist.github.com/plentz/6737338 → Best nginx configuration for improved security(and performance). Acesso em 25/03.
[99] https://blog.streamroot.io/encode-multi-bitrate-videos-mpeg-dash-mse-based-media-players-22/ → HOW TO ENCODE MULTI-BITRATE VIDEOS IN MPEG-DASH FOR MSE BASED MEDIA PLAYERS (2/2). Acesso em 28/03.