ADVANCE MULTIMEDIA SERVICES

COURSE 2025

PRACTICE 2

Application testing and exchange of information
(*MultiAdvancedFlow*)

Alberto López Casanova
Manuel Ángel Vega Gómez

INDEX

INDEX OF FIGURES

1 Introduction

In this section, we will test our multimedia servers, focusing on two notable cases. The first involves uploading a video to our streaming platform and then playing it back — in other words, we will perform video on demand. In the second case, we will use a mobile phone with the well-known Larix Broadcaster app to simulate real-time video streaming. Finally, to push our device to its limits, we will establish two sessions on our server to implement both previous cases simultaneously

Through this analysis, we Will be able to know how our application will respond under different networking conditions and configurations.

1.1 Tools for testing

Nowadays, we have many tools available to accurately measure our network environments. In this specific case, we have chosen to use the Node Server's own application, as well as Wireshark.

Under idle conditions, our server shows the following characteristics across its various components.

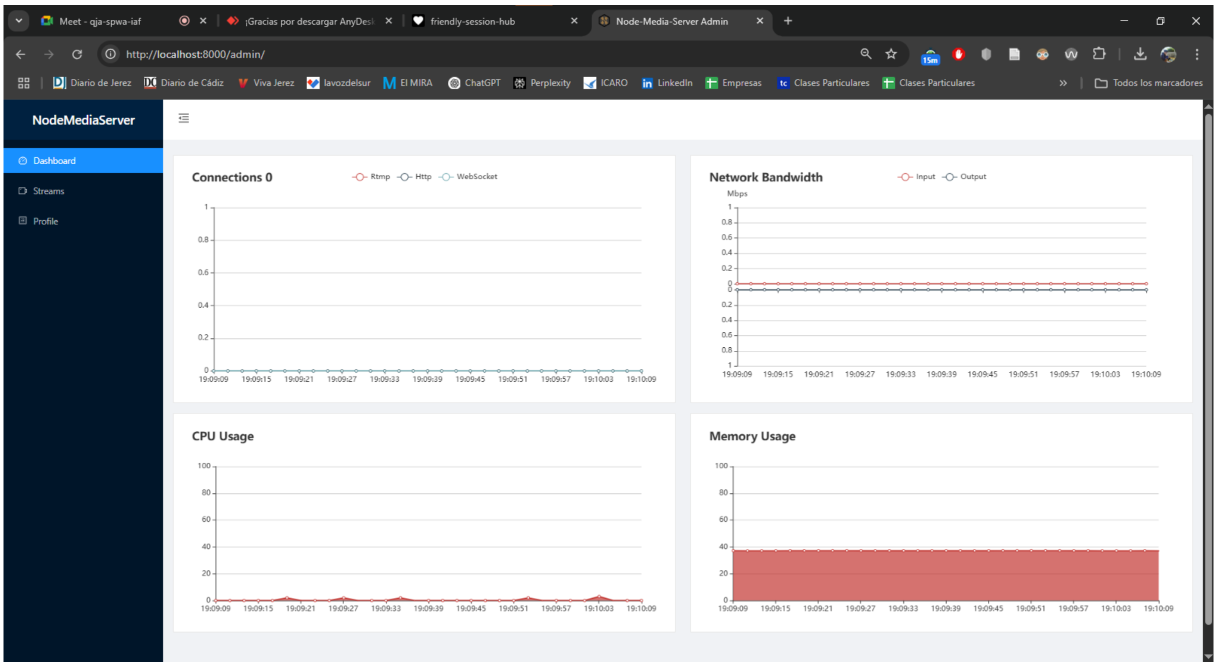
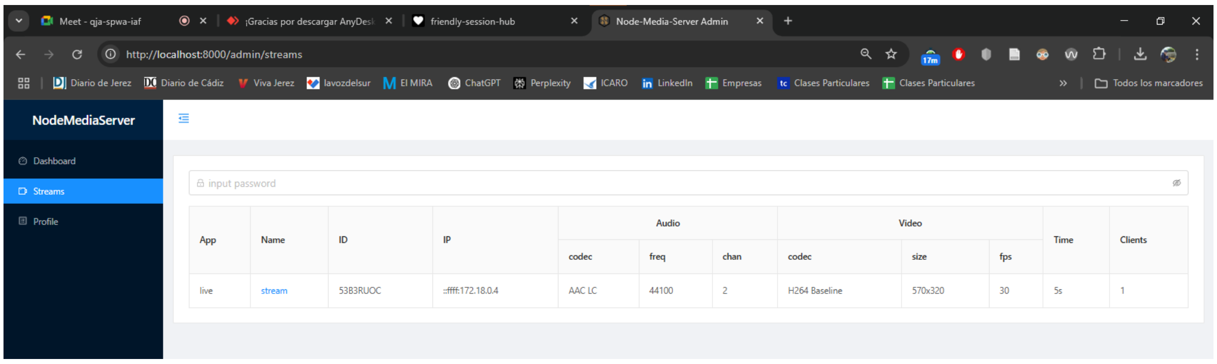


Figure 1-1 server statistics

2 Video On-demand

Frist of all, we must transmit a new video to make a stream in order to check our video on-demand server



App	Name	ID	IP	Audio			Video			Time	Clients
				codec	freq	chan	codec	size	fps		
live	stream	53B3RUOC	::ffff:172.18.0.4	AAC LC	44100	2	H264 Baseline	570x320	30	5s	1

Figure 2-1 streams available on the server

As a result, when checking the "streams" section on the Node Media Server web page, we can see that a live stream named "stream" has been created — this is the name we assigned to our stream.

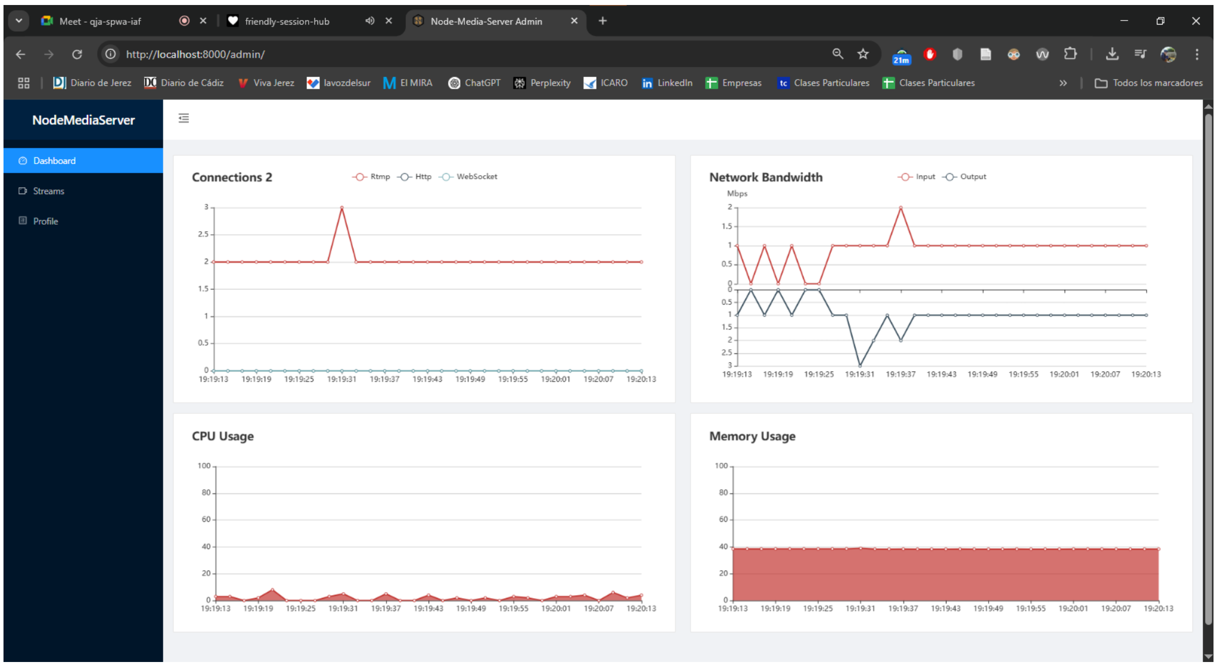
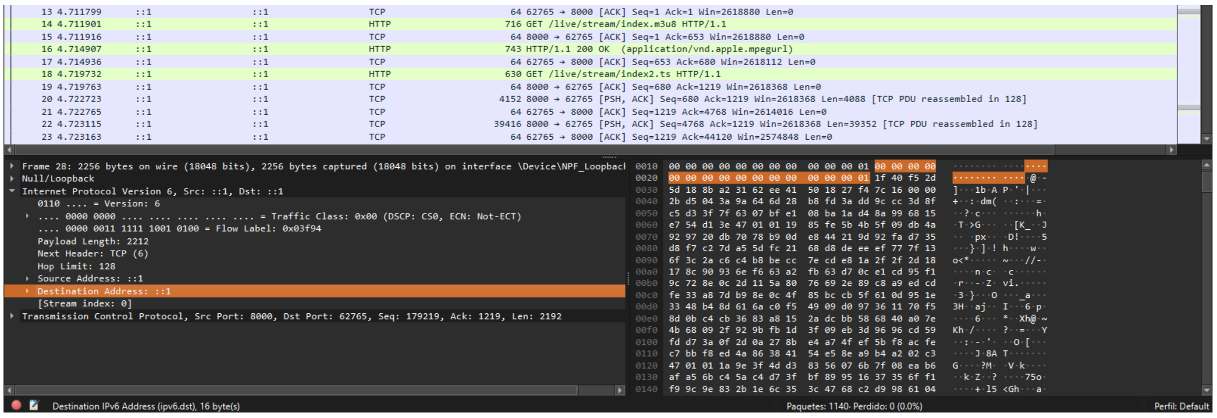


Figure 2-2 server statistics

It is also noticeable that there are two video channels: one for the incoming stream and another for the default outgoing stream. On the server's main page, we can see two active connections, corresponding to the two channels mentioned earlier.

Using Wireshark, we confirmed that all HTTP requests sent to the server were successfully completed. Another thing to consider is when the communication just started, we can see all the sequences of the stream appear on the capture



```
   13 4.711799    ::1            ::1            TCP          64 62765 → 8000 [ACK] Seq=1 Ack=1 Win=2618880 Len=0
   14 4.711901    ::1            ::1            HTTP        716 GET /live/stream/index.m3u8 HTTP/1.1
   15 4.711916    ::1            ::1            TCP          64 8000 → 62765 [ACK] Seq=1 Ack=653 Win=2618880 Len=0
   16 4.714907    ::1            ::1            HTTP        743 HTTP/1.1 200 OK  (application/vnd.apple.mpegurl)
   17 4.714936    ::1            ::1            TCP          64 62765 → 8000 [ACK] Seq=653 Ack=680 Win=2618112 Len=0
   18 4.719732    ::1            ::1            HTTP        630 GET /live/stream/index2.ts HTTP/1.1
   19 4.719763    ::1            ::1            TCP          64 8000 → 62765 [ACK] Seq=680 Ack=1219 Win=2618368 Len=0
   20 4.722723    ::1            ::1            TCP        4152 8000 → 62765 [PSH, ACK] Seq=680 Ack=1219 Win=2618368 Len=4088 [TCP PDU reassembled in 128]
   21 4.722765    ::1            ::1            TCP          64 62765 → 8000 [ACK] Seq=1219 Ack=4768 Win=2614016 Len=0
   22 4.723115    ::1            ::1            TCP       39416 8000 → 62765 [PSH, ACK] Seq=4768 Ack=1219 Win=2618368 Len=39352 [TCP PDU reassembled in 128]
   23 4.723163    ::1            ::1            TCP          64 62765 → 8000 [ACK] Seq=1219 Ack=44120 Win=2574848 Len=0
```

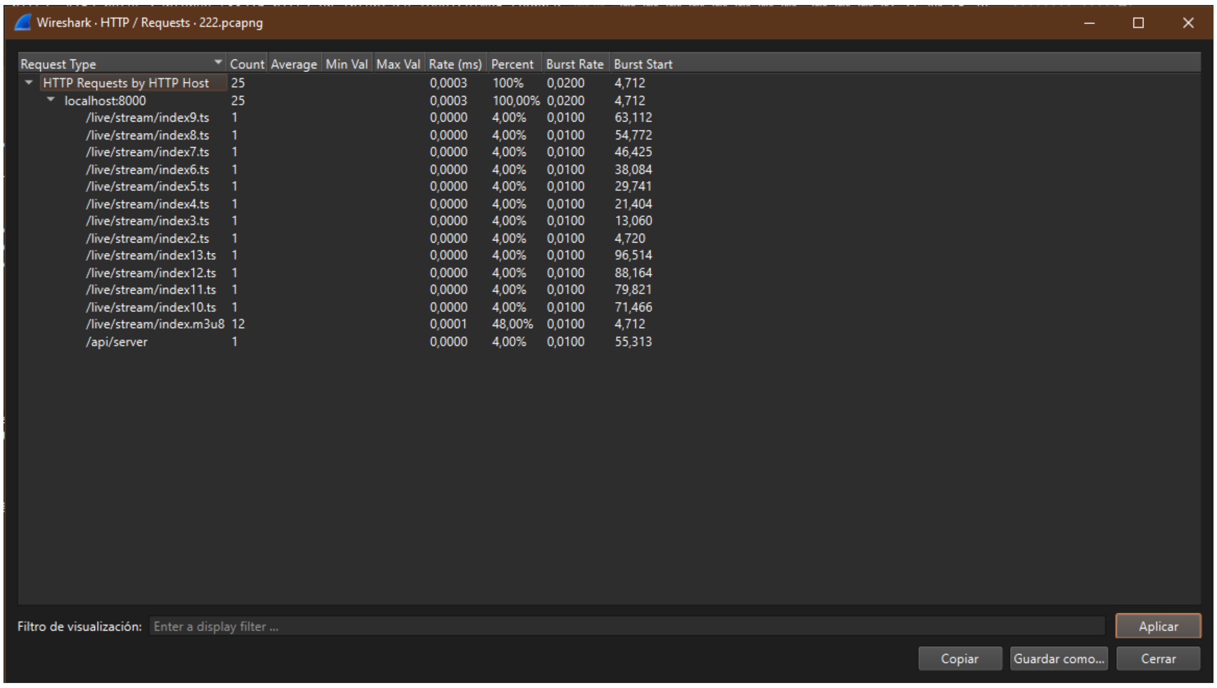
```
> Frame 28: 2256 bytes on wire (18048 bits), 2256 bytes captured (18048 bits) on interface \Device\NPF_Loopbacl
> Null/Loopback
v Internet Protocol Version 6, Src: ::1, Dst: ::1
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0011 1111 1001 0100 = Flow Label: 0x03f94
    Payload Length: 2212
    Next Header: TCP (6)
    Hop Limit: 128
  > Source Address: ::1
  > Destination Address: ::1
    [Stream index: 0]
> Transmission Control Protocol, Src Port: 8000, Dst Port: 62765, Seq: 179219, Ack: 1219, Len: 2192
```

```
0010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ········ ········
0020  00 00 00 00 00 00 00 00  00 00 00 01 1f 40 f5 2d   ········ ····@·-
0030  5d 18 8b a2 31 62 ee 41  50 18 27 f4 7c 16 00 00   ]···1b·A P·'·|···
0040  2b d5 04 3a 9a 64 6d 28  b8 fd 3a dd 9c cc 3d 8f   +··:·dm( ··:···=·
0050  c5 d3 3f 7f 63 07 bf e1  08 ba 1a d4 8a 99 68 15   ··?·c··· ······h·
0060  e7 54 d1 3e 47 01 01 19  85 fe 5b 4b 5f 09 db 4a   ·T·>G··· ··[K_··J
0070  92 97 20 db 70 78 b9 0d  e8 44 21 9d 92 fa d7 35   ·· ·px·· ·D!····5
0080  d8 f7 c2 7d a5 5d fc 21  68 d8 de ee ef 77 7f 13   ···}·]·! h····w··
0090  6f 3c 2a c6 c4 b8 be cc  7e cd e8 1a 2f 2f 2d 18   o<*····· ~···//-·
00a0  17 8c 90 93 6e f6 63 a2  fb 63 d7 0c e1 cd 95 f1   ····n·c· ·c······
00b0  9c 72 8e 0c 2d 11 5a 80  76 69 2e 89 c8 a9 ed cd   ·r··-·Z· vi.·····
00c0  fe 33 a8 7d b9 8e 0c 4f  85 bc cb 5f 61 0d 95 1e   ·3·}···O ···_a···
00d0  33 48 b4 8d 61 6a c0 f5  49 09 d0 97 36 11 70 f5   3H··aj·· I···6·p·
00e0  8d 0b c4 cb 36 83 a8 15  2a dc bb 58 68 40 a0 7e   ····6··· *··Xh@·~
00f0  4b 68 09 2f 92 9b fb 1d  3f 09 eb 3d 96 96 cd 59   Kh·/···· ?··=···Y
0100  fd d7 3a 0f 2d 0a 27 8b  e4 a7 4f ef 5b f8 ac fe   ··:·-·'· ··O·[···
0110  c7 bb f8 ed 4a 86 38 41  54 e5 8e a9 b4 a2 02 c3   ····J·8A T·······
0120  47 01 01 1a 9e 3f 4d d3  83 56 07 6b 7f 08 ea b6   G····?M· ·V·k····
0130  af a5 6b c4 5a c4 d7 3f  bf 89 95 16 37 35 6f f1   ··k·Z··? ····75o·
0140  f9 9c 9e 83 2b 1e 6c 35  3c 47 68 c2 d9 98 61 04   ····+·l5 <Gh···a·
```

```
Destination IPv6 Address (ipv6.dst), 16 byte(s)        Paquetes: 1140· Perdido: 0 (0.0%)        Perfil: Default
```

Figure 2-3 successful communication between client and server

If want go to in deeper in this point we can choose to count the http petitions during the whole capture



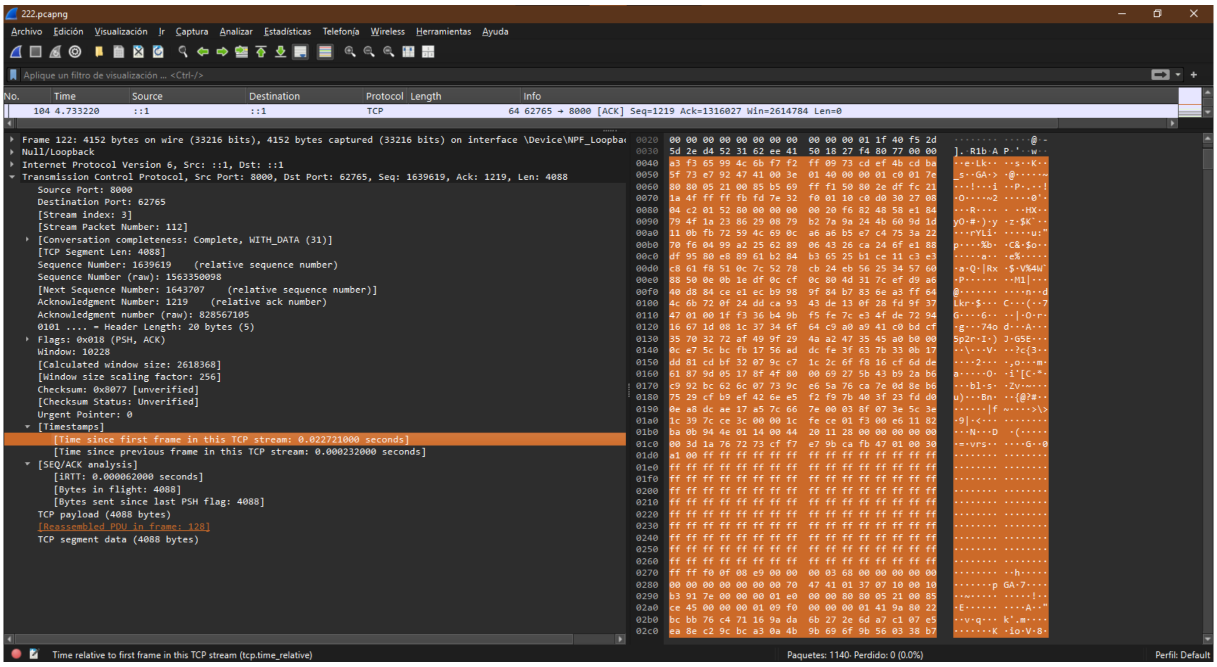
Wireshark · HTTP / Requests · 222.pcapng

Request Type	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
HTTP Requests by HTTP Host	25				0,0003	100%	0,0200	4,712
localhost:8000	25				0,0003	100,00%	0,0200	4,712
/live/stream/index9.ts	1				0,0000	4,00%	0,0100	63,112
/live/stream/index8.ts	1				0,0000	4,00%	0,0100	54,772
/live/stream/index7.ts	1				0,0000	4,00%	0,0100	46,425
/live/stream/index6.ts	1				0,0000	4,00%	0,0100	38,084
/live/stream/index5.ts	1				0,0000	4,00%	0,0100	29,741
/live/stream/index4.ts	1				0,0000	4,00%	0,0100	21,404
/live/stream/index3.ts	1				0,0000	4,00%	0,0100	13,060
/live/stream/index2.ts	1				0,0000	4,00%	0,0100	4,720
/live/stream/index13.ts	1				0,0000	4,00%	0,0100	96,514
/live/stream/index12.ts	1				0,0000	4,00%	0,0100	88,164
/live/stream/index11.ts	1				0,0000	4,00%	0,0100	79,821
/live/stream/index10.ts	1				0,0000	4,00%	0,0100	71,466
/live/stream/index.m3u8	12				0,0001	48,00%	0,0100	4,712
/api/server	1				0,0000	4,00%	0,0100	55,313

Filtro de visualización: Enter a display filter ...

Figure 2-4 http requests

From the figure below, we can observe that the stream reception delay is 20 microseconds, which is relatively low since both the streaming server and the client are running on the same machine.


222.pcapng
Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda
Aplique un filtro de visualización … <Ctrl-/>

```
No.   Time      Source    Destination   Protocol Length  Info
  104 4.733220  ::1       ::1           TCP          64  62765 → 8000 [ACK] Seq=1219 Ack=1316027 Win=2614784 Len=0
```

```
› Frame 122: 4152 bytes on wire (33216 bits), 4152 bytes captured (33216 bits) on interface \Device\NPF_Loopbac
› Null/Loopback
› Internet Protocol Version 6, Src: ::1, Dst: ::1
˅ Transmission Control Protocol, Src Port: 8000, Dst Port: 62765, Seq: 1639619, Ack: 1219, Len: 4088
    Source Port: 8000
    Destination Port: 62765
    [Stream index: 3]
    [Stream Packet Number: 112]
  › [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 4088]
    Sequence Number: 1639619    (relative sequence number)
    Sequence Number (raw): 1563350098
    [Next Sequence Number: 1643707    (relative sequence number)]
    Acknowledgment Number: 1219    (relative ack number)
    Acknowledgment number (raw): 828567105
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x018 (PSH, ACK)
    Window: 10228
    [Calculated window size: 2618368]
    [Window size scaling factor: 256]
    Checksum: 0x8077 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ˅ [Timestamps]
       [Time since first frame in this TCP stream: 0.022721000 seconds]
       [Time since previous frame in this TCP stream: 0.000232000 seconds]
  ˅ [SEQ/ACK analysis]
       [iRTT: 0.000062000 seconds]
       [Bytes in flight: 4088]
       [Bytes sent since last PSH flag: 4088]
    TCP payload (4088 bytes)
    [Reassembled PDU in frame: 128]
    TCP segment data (4088 bytes)
```

```
0020  00 00 00 00 00 00 00 00  00 00 00 01 1f 40 f5 2d   ········ ·····@·-
0030  5d 2e d4 52 31 62 ee 41  50 18 27 f4 80 77 00 00   ].·R1b·A P·'··w··
0040  a3 f3 65 99 4c 6b f7 f2  ff 09 73 cd ef 4b cd ba   ··e·Lk·· ··s··K··
0050  5f 73 e7 92 47 41 00 3e  01 40 00 00 01 c0 01 7e   _s··GA·> ·@·····~
0060  80 80 05 21 00 85 b5 69  ff f1 50 80 2e df fc 21   ···!···i ··P·.··!
0070  1a 4f ff ff fb fd 7e 32  f0 01 10 c0 d0 30 27 08   ·O····~2 ·····0'·
0080  04 c2 01 52 80 00 00 00  00 20 f6 82 48 58 e1 84   ···R···· · ··HX··
0090  79 4f 1a 23 86 29 08 79  b2 7a 9a 24 4b 60 9d 1d   yO·#·)·y ·z·$K`··
00a0  11 0b fb 72 59 4c 69 0c  a6 a6 b5 e7 c4 75 3a 22   ···rYLi· ·····u:"
00b0  70 f6 04 99 a2 25 62 89  06 43 26 ca 24 6f e1 88   p····%b· ·C&·$o··
00c0  df 95 80 e8 89 61 b2 84  b3 65 25 b1 ce 11 c3 e3   ·····a·· ·e%·····
00d0  c8 61 f8 51 0c 7c 52 78  cb 24 eb 56 25 34 57 60   ·a·Q·|Rx ·$·V%4W`
00e0  88 50 0e 0b 1e df 0c cf  0c 80 4d 31 7c ef d9 a6   ·P······ ··M1|···
00f0  40 d8 84 ce e1 ec b9 98  9f 84 b7 83 6e a3 ff 64   @······· ····n··d
0100  4c 6b 72 0f 24 dd ca 93  43 de 13 0f 28 fd 9f 37   Lkr·$··· C···(··7
0110  47 01 00 1f f3 36 b4 9b  f5 fe 7c e3 4f de 72 94   G····6·· ··|·O·r·
0120  16 67 1d 08 1c 37 34 6f  64 c9 a0 a9 41 c0 bd cf   ·g···74o d···A···
0130  35 70 32 72 af 49 9f 29  4a a2 47 35 45 a0 b0 00   5p2r·I·) J·G5E···
0140  0c e7 5c bc fb 17 56 ad  dc fe 3f 63 7b 33 0b 17   ··\···V· ··?c{3··
0150  dd 81 cd bf 32 07 9c c7  1c 2c 6f f8 16 cf 6d de   ····2··· ·,o···m·
0160  61 87 9d 05 17 8f 4f 80  00 69 27 5b 43 b9 2a b6   a·····O· ·i'[C·*·
0170  c9 92 bc 62 6c 07 73 9c  e6 5a 76 ca 7e 0d 8e b6   ···bl·s· ·Zv·~···
0180  75 29 cf b9 ef 42 6e e5  f2 f9 7b 40 3f 23 fd d0   u)···Bn· ··{@?#··
0190  0e a8 dc ae 17 a5 7c 66  7e 00 03 8f 07 3e 5c 3e   ······|f ~····>\>
01a0  1c 39 7c ce 3c 00 00 1c  fe ce 01 f3 00 e6 11 82   ·9|·<··· ········
01b0  ba 0b 94 4e 01 14 00 44  20 11 28 00 00 00 00 00   ···N···D  ·(·····
01c0  00 3d 1a 76 72 73 cf f7  e7 9b ca fb 47 01 00 30   ·=·vrs·· ····G··0
01d0  a1 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
01e0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
01f0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0200  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0210  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0220  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0230  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0240  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0250  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0260  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ········ ········
0270  ff ff f0 0f 08 e9 00 00  00 03 68 00 00 00 00 00   ········ ··h·····
0280  00 00 00 00 00 00 00 70  47 41 01 37 07 10 00 10   ·······p GA·7····
0290  b3 91 7e 00 00 00 01 e0  00 00 80 80 05 21 00 85   ··~····· ·····!··
02a0  ce 45 00 00 00 01 09 f0  00 00 00 01 41 9a 80 22   ·E······ ····A··"
02b0  bc bb 76 c4 71 16 9a da  6b 27 2e 6d a7 c1 07 e5   ··v·q··· k'.m····
02c0  ea 8e c2 9c bc a3 0a 4b  9b 69 6f 9b 56 03 38 b7   ·······K ·io·V·8·
```

```
Time relative to first frame in this TCP stream (tcp.time_relative)      Paquetes: 1140· Perdido: 0 (0.0%)      Perfil: Default
```

Figure 2-5 packet delay

From this point forward, our analysis will focus on scenarios involving the default FLV stream provided by NMS (Node Media Server). These scenarios involve accessing the FLV stream through HTTP requests, allowing us to explore its features and potential applications in greater depth.

3 Real time streaming

In the next phase of our project, we will simulate a real-time streaming environment by using the Larix Broadcaster application on a mobile device. This setup allows us to emulate a realistic live streaming scenario where the video content is captured and transmitted in real time over the network.

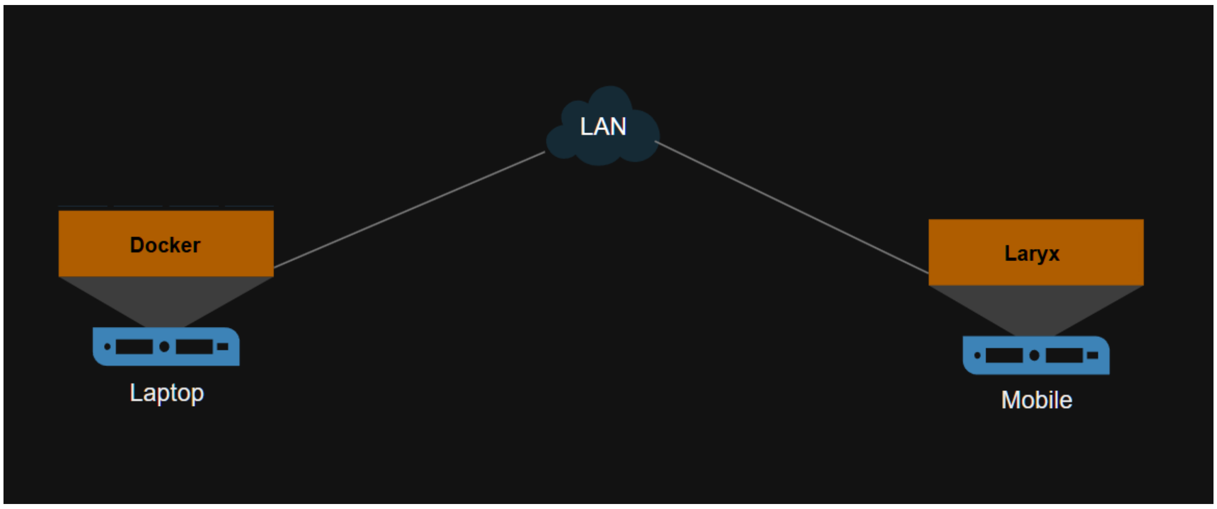
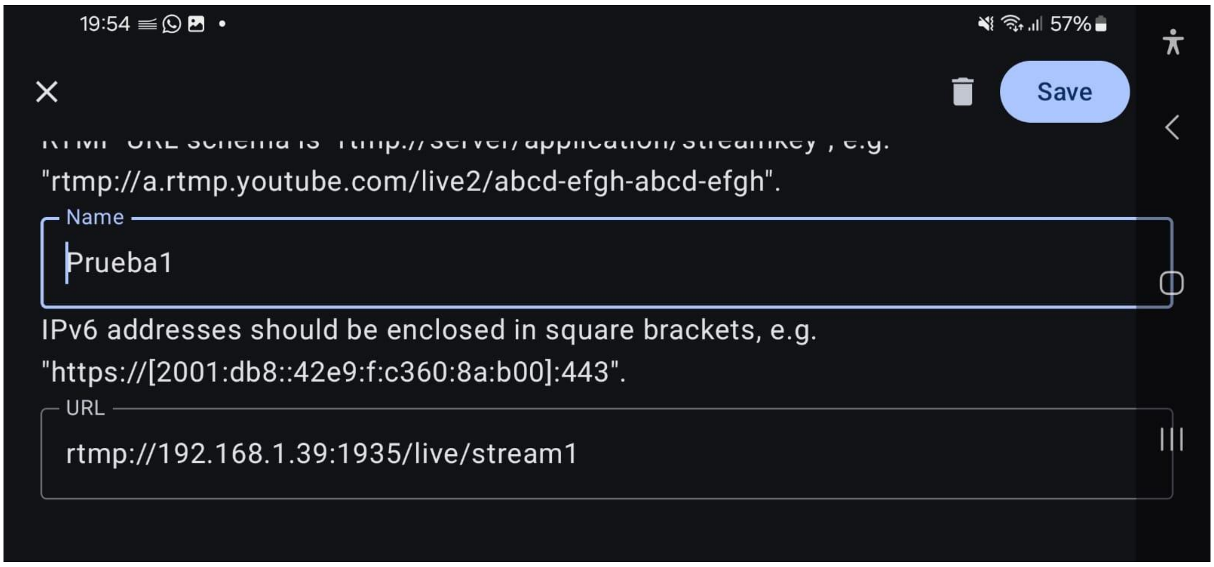


Figure 3-1 Scenario of real time streaming

The mobile phone will act as the source, streaming live video directly to our Node Media Server (NMS) instance. This simulation is crucial for evaluating the behaviour, stability, and performance of the server under conditions that closely resemble real-world usage, where latency, network fluctuations, and device performance can all influence the overall streaming experience. Through this test, we aim to observe how efficiently the server handles incoming live streams, how it manages resources when receiving data from an external device, and how responsive it remains when serving the content to potential viewers. Additionally, capturing and analysing the traffic with Wireshark will provide further insights into protocol behaviour, network load, and the quality of the live transmission.

The first step will be to configure the Larix Broadcaster application on our mobile phone so that it can transmit the video stream to the IP address of the PC where our server is running. During the configuration process, it is important to assign a new, unique name to our live stream.



"rtmp://a.rtmp.youtube.com/live2/abcd-efgh-abcd-efgh".

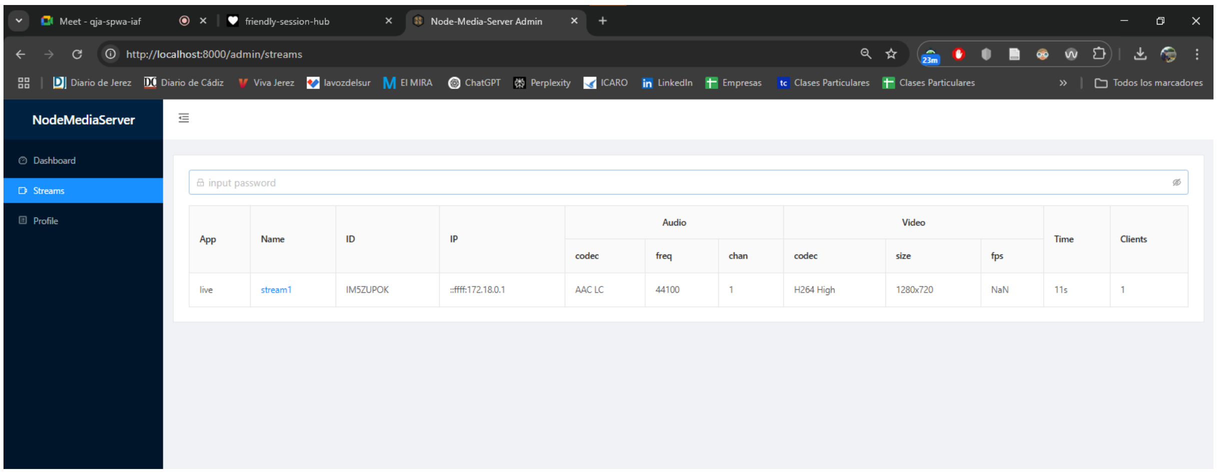
Name: Prueba1

IPv6 addresses should be enclosed in square brackets, e.g. "https://[2001:db8::42e9:f:c360:8a:b00]:443".

URL: rtmp://192.168.1.39:1935/live/stream1

Figure 3-2 configuration of Larix Broadcaster

We must ensure that it is different from any previously used stream names to avoid conflicts on the server. If the same name were reused, it could cause errors or unstable behaviour during the session.



http://localhost:8000/admin/streams

App	Name	ID	IP	Audio			Video			Time	Clients
				codec	freq	chan	codec	size	fps		
live	stream1	IM5ZUPOK	::ffff:172.18.0.1	AAC LC	44100	1	H264 High	1280x720	NaN	11s	1

Figure 3-3 streams available on the server

Once the stream is set up and transmission begins, we can monitor the server statistics through the Node Media Server web interface. The server's statistics reflect the current state of the stream, including active connections, bitrate, and other performance indicators.

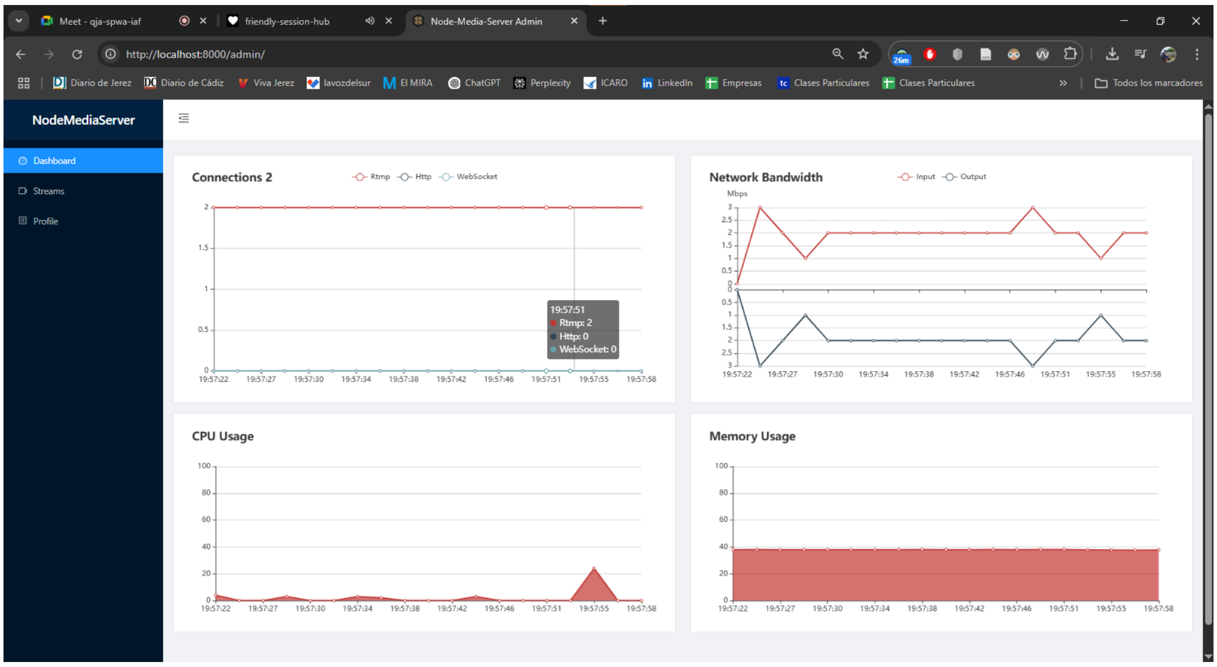
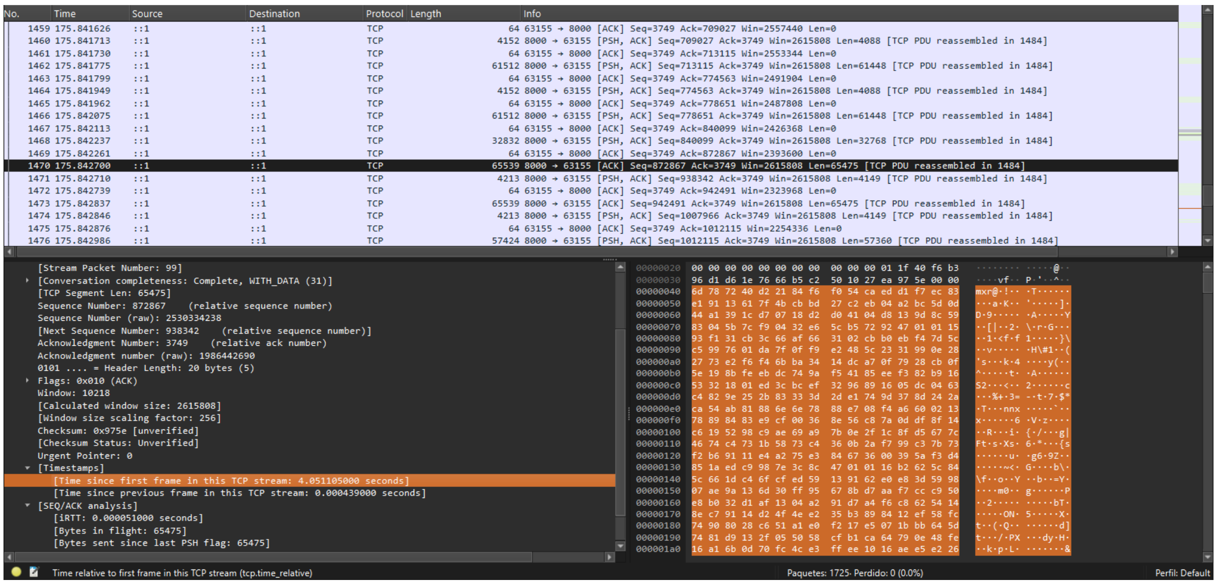


Figure 3-4 server statistics

Another important observation is the latency of the transmission. In this setup, we recorded the delay. This delay can be clearly seen in the figures provided below and is typical for real-time streaming



```
No.  Time       Source  Destination  Protocol Length  Info
1459 175.841626 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=709027 Win=2557440 Len=0
1460 175.841713 ::1     ::1          TCP      4152 8000 → 63155 [PSH, ACK] Seq=709027 Ack=3749 Win=2615808 Len=4088 [TCP PDU reassembled in 1484]
1461 175.841730 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=713115 Win=2553344 Len=0
1462 175.841775 ::1     ::1          TCP      61512 8000 → 63155 [PSH, ACK] Seq=713115 Ack=3749 Win=2615808 Len=61448 [TCP PDU reassembled in 1484]
1463 175.841799 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=774563 Win=2491904 Len=0
1464 175.841949 ::1     ::1          TCP      4152 8000 → 63155 [PSH, ACK] Seq=774563 Ack=3749 Win=2615808 Len=4088 [TCP PDU reassembled in 1484]
1465 175.841962 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=778651 Win=2487808 Len=0
1466 175.842075 ::1     ::1          TCP      61512 8000 → 63155 [PSH, ACK] Seq=778651 Ack=3749 Win=2615808 Len=61448 [TCP PDU reassembled in 1484]
1467 175.842113 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=840099 Win=2426368 Len=0
1468 175.842237 ::1     ::1          TCP      32832 8000 → 63155 [PSH, ACK] Seq=840099 Ack=3749 Win=2615808 Len=32768 [TCP PDU reassembled in 1484]
1469 175.842261 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=872867 Win=2393600 Len=0
1470 175.842700 ::1     ::1          TCP      65539 8000 → 63155 [ACK] Seq=872867 Ack=3749 Win=2615808 Len=65475 [TCP PDU reassembled in 1484]
1471 175.842710 ::1     ::1          TCP      4213 8000 → 63155 [PSH, ACK] Seq=938342 Ack=3749 Win=2615808 Len=4149 [TCP PDU reassembled in 1484]
1472 175.842739 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=942491 Win=2323968 Len=0
1473 175.842837 ::1     ::1          TCP      65539 8000 → 63155 [ACK] Seq=942491 Ack=3749 Win=2615808 Len=65475 [TCP PDU reassembled in 1484]
1474 175.842846 ::1     ::1          TCP      4213 8000 → 63155 [PSH, ACK] Seq=1007966 Ack=3749 Win=2615808 Len=4149 [TCP PDU reassembled in 1484]
1475 175.842876 ::1     ::1          TCP      64 63155 → 8000 [ACK] Seq=3749 Ack=1012115 Win=2254336 Len=0
1476 175.842986 ::1     ::1          TCP      57424 8000 → 63155 [PSH, ACK] Seq=1012115 Ack=3749 Win=2615808 Len=57360 [TCP PDU reassembled in 1484]
```

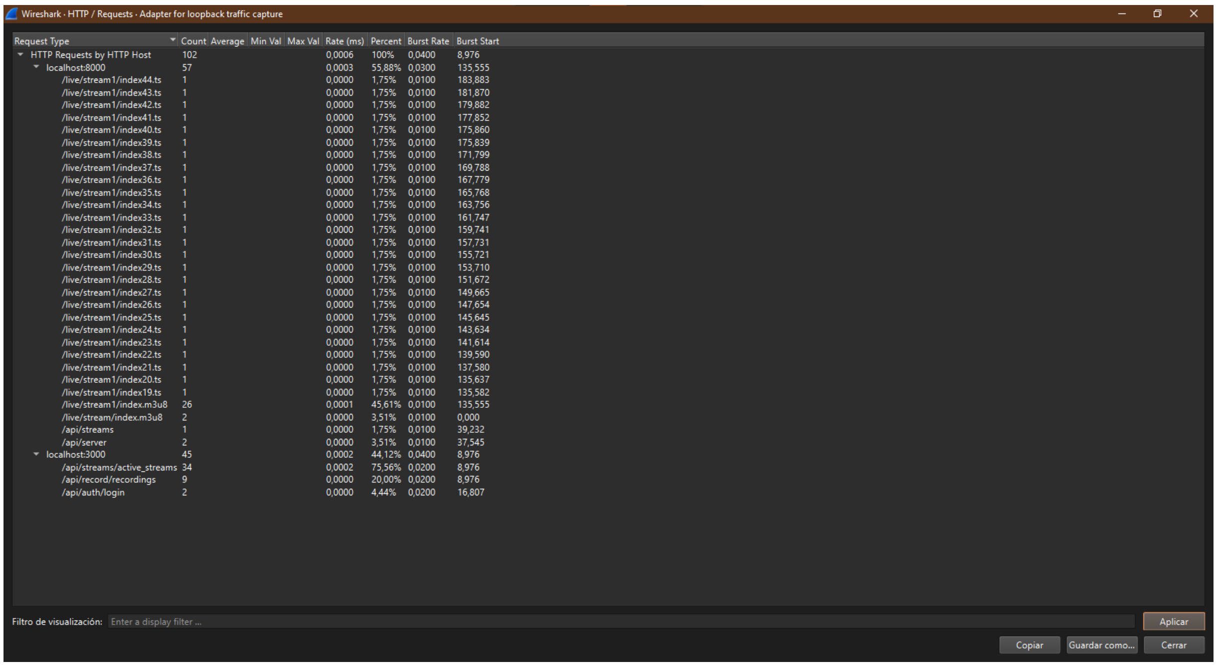
```
    [Stream Packet Number: 99]
  › [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 65475]
    Sequence Number: 872867    (relative sequence number)
    Sequence Number (raw): 2530334238
    [Next Sequence Number: 938342    (relative sequence number)]
    Acknowledgment Number: 3749    (relative ack number)
    Acknowledgment number (raw): 1986442690
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x010 (ACK)
    Window: 10218
    [Calculated window size: 2615808]
    [Window size scaling factor: 256]
    Checksum: 0x975e [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ˅ [Timestamps]
       [Time since first frame in this TCP stream: 4.051105000 seconds]
       [Time since previous frame in this TCP stream: 0.000439000 seconds]
  ˅ [SEQ/ACK analysis]
       [iRTT: 0.000051000 seconds]
       [Bytes in flight: 65475]
       [Bytes sent since last PSH flag: 65475]
```

```
00000020  00 00 00 00 00 00 00 00  00 00 00 01 1f 40 f6 b3   ········ ·····@··
00000030  96 d1 d6 1e 76 66 b5 c2  50 10 27 ea 97 5e 00 00   ····vf·· P·'··^··
00000040  6d 78 72 40 d2 21 84 f6  f0 54 ca ed d1 f7 ec 83   mxr@·!·· ·T······
00000050  e1 91 13 61 7f 4b cb bd  27 c2 eb 04 a2 bc 5d 0d   ···a·K·· '·····]·
00000060  44 a1 39 1c d7 07 18 d2  d0 41 04 d8 13 9d 8c 59   D·9····· ·A·····Y
00000070  83 04 5b 7c f9 04 32 e6  5c b5 72 92 47 01 01 15   ··[|··2· \·r·G···
00000080  93 f1 31 cb 3c 66 af 66  31 02 cb b0 eb f4 7d 5c   ··1·<f·f 1·····}\
00000090  c5 99 76 01 da 7f 0f f9  e2 48 5c 23 31 99 0e 28   ··v····· ·H\#1··(
000000a0  27 73 e2 f6 f4 6b ba 34  14 dc a7 0f 79 28 cb 0f   's···k·4 ····y(··
000000b0  5e 19 8b fe eb dc 74 9a  f5 41 85 ee f3 82 b9 16   ^·····t· ·A······
000000c0  53 32 18 01 ed 3c bc ef  32 96 89 16 05 dc 04 63   S2···<·· 2······c
000000d0  c4 82 9e 25 2b 83 33 3d  2d e1 74 9d 37 8d 24 2a   ···%+·3= -·t·7·$*
000000e0  ca 54 ab 81 88 6e 6e 78  88 e7 08 f4 a6 60 02 13   ·T···nnx ·····`··
000000f0  78 89 84 83 e9 cf 00 36  8e 56 c8 7a 0d df 8f 14   x······6 ·V·z····
00000100  c6 19 52 98 c9 ae 69 a9  7b 0e 2f 1c 8f d5 67 7c   ··R···i· {·/···g|
00000110  46 74 c4 73 1b 58 73 c4  36 0b 2a f7 99 c3 7b 73   Ft·s·Xs· 6·*···{s
00000120  f2 b6 91 11 e4 a2 75 e3  84 67 36 00 39 5a f3 d4   ······u· ·g6·9Z··
00000130  85 1a ed c9 98 7e 3c 8c  47 01 01 16 b2 62 5c 84   ·····~<· G····b\·
00000140  5c 66 1d c4 6f cf ed 59  13 91 62 e0 e8 3d 59 98   \f··o··Y ··b··=Y·
00000150  07 ae 9a 13 6d 30 ff 95  67 8b d7 aa f7 cc c9 50   ····m0·· g······P
00000160  e8 b0 32 d1 af 13 04 a2  91 d7 a4 f6 c8 62 54 14   ··2····· ·····bT·
00000170  8e c7 91 14 d2 4f 4e e2  35 b3 89 84 12 ef 58 fc   ·····ON· 5·····X·
00000180  74 90 80 28 c6 51 a1 e0  f2 17 e5 07 1b bb 64 5d   t··(·Q·· ······d]
00000190  74 81 d9 13 2f 05 50 58  cf b1 ca 64 79 0e 48 fe   t···/·PX ···dy·H·
000001a0  16 a1 6b 0d 70 fc 4c e3  ff ee 10 16 ae e5 e2 26   ··k·p·L· ·······&
```

```
Time relative to first frame in this TCP stream (tcp.time_relative)     Paquetes: 1725· Perdido: 0 (0.0%)     Perfil: Default
```

Figure 3-5 packets delay

In this real-time streaming scenario, the number of HTTP requests to the server significantly increases compared to previous video-on-demand tests. This rise in requests is mainly because we are now dealing with a continuous live feed, and, additionally, the server and client are operating on separate devices, leading to more network activity.



Wireshark · HTTP / Requests · Adapter for loopback traffic capture

Request Type	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
HTTP Requests by HTTP Host	102				0,0006	100%	0,0400	8,976
localhost:8000	57				0,0003	55,88%	0,0300	135,555
/live/stream1/index44.ts	1				0,0000	1,75%	0,0100	183,883
/live/stream1/index43.ts	1				0,0000	1,75%	0,0100	181,870
/live/stream1/index42.ts	1				0,0000	1,75%	0,0100	179,882
/live/stream1/index41.ts	1				0,0000	1,75%	0,0100	177,852
/live/stream1/index40.ts	1				0,0000	1,75%	0,0100	175,860
/live/stream1/index39.ts	1				0,0000	1,75%	0,0100	175,839
/live/stream1/index38.ts	1				0,0000	1,75%	0,0100	171,799
/live/stream1/index37.ts	1				0,0000	1,75%	0,0100	169,788
/live/stream1/index36.ts	1				0,0000	1,75%	0,0100	167,779
/live/stream1/index35.ts	1				0,0000	1,75%	0,0100	165,768
/live/stream1/index34.ts	1				0,0000	1,75%	0,0100	163,756
/live/stream1/index33.ts	1				0,0000	1,75%	0,0100	161,747
/live/stream1/index32.ts	1				0,0000	1,75%	0,0100	159,741
/live/stream1/index31.ts	1				0,0000	1,75%	0,0100	157,731
/live/stream1/index30.ts	1				0,0000	1,75%	0,0100	155,721
/live/stream1/index29.ts	1				0,0000	1,75%	0,0100	153,710
/live/stream1/index28.ts	1				0,0000	1,75%	0,0100	151,672
/live/stream1/index27.ts	1				0,0000	1,75%	0,0100	149,665
/live/stream1/index26.ts	1				0,0000	1,75%	0,0100	147,654
/live/stream1/index25.ts	1				0,0000	1,75%	0,0100	145,645
/live/stream1/index24.ts	1				0,0000	1,75%	0,0100	143,634
/live/stream1/index23.ts	1				0,0000	1,75%	0,0100	141,614
/live/stream1/index22.ts	1				0,0000	1,75%	0,0100	139,590
/live/stream1/index21.ts	1				0,0000	1,75%	0,0100	137,580
/live/stream1/index20.ts	1				0,0000	1,75%	0,0100	135,637
/live/stream1/index19.ts	1				0,0000	1,75%	0,0100	135,582
/live/stream1/index.m3u8	26				0,0001	45,61%	0,0100	135,555
/live/stream/index.m3u8	2				0,0000	3,51%	0,0100	0,000
/api/streams	1				0,0000	1,75%	0,0100	39,232
/api/server	2				0,0000	3,51%	0,0100	37,545
localhost:3000	45				0,0002	44,12%	0,0400	8,976
/api/streams/active_streams	34				0,0002	75,56%	0,0200	8,976
/api/record/recordings	9				0,0000	20,00%	0,0200	8,976
/api/auth/login	2				0,0000	4,44%	0,0200	16,807

Figure 3-6 http requests

4 Real time streaming WAN

In the next stage of our analysis, we will focus on a more complex scenario where the streaming process will take place across different networks. To achieve this, we will set up a secure tunnel to connect both networks, allowing the mobile device and the server to communicate as if they were on the same local network. This approach simulates a more realistic situation where the broadcaster and the streaming server are not within the same local infrastructure, which is common in real-world applications.

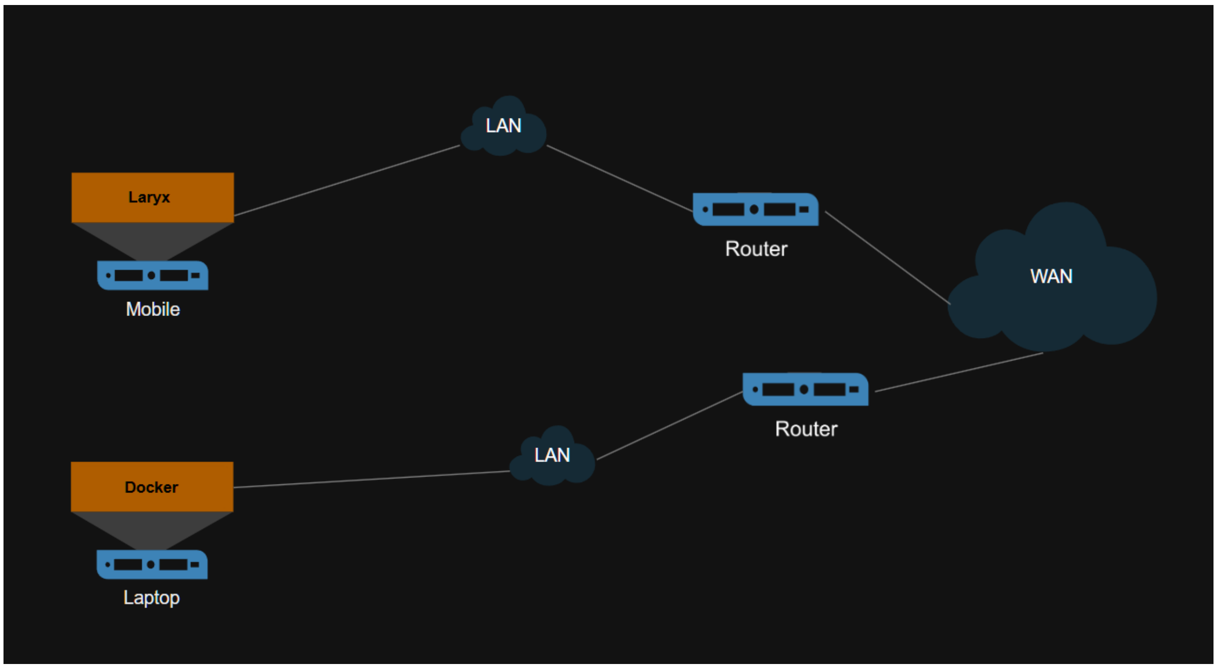
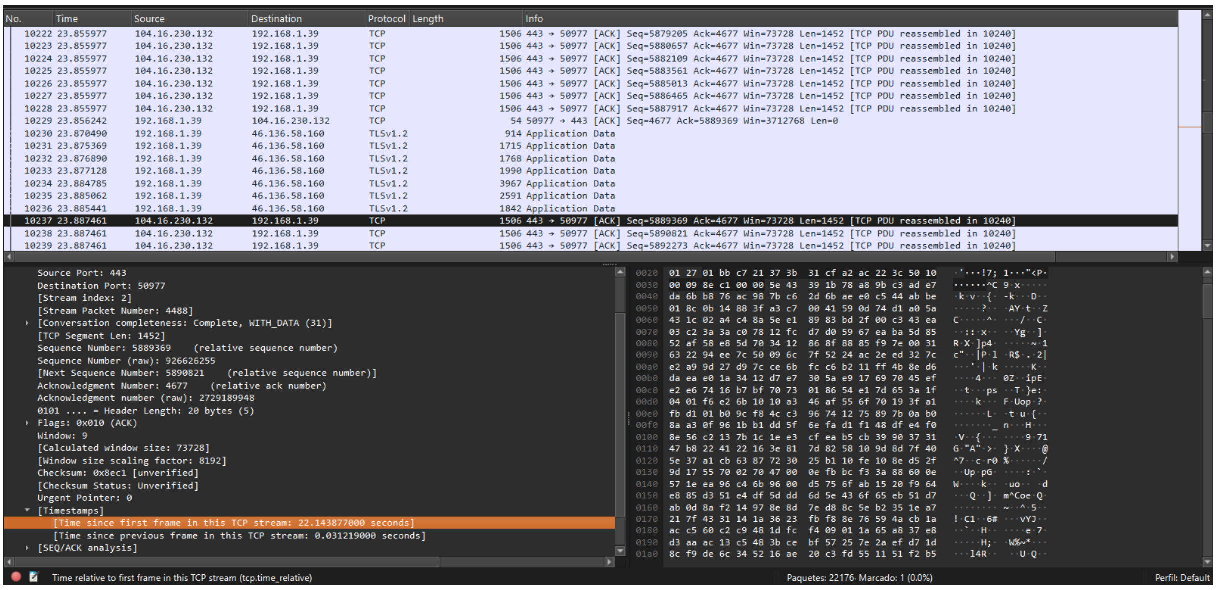


Figure 4-1 Scenario of real time streaming on WAN

As before, we will configure the Larix Broadcaster application to send the live stream, this time directing it to the server's IP address through the tunnel. We will again assign a unique name to the live transmission to avoid conflicts with previous sessions that could cause instability or connection issues.

By using separate networks connected through a tunnel, we anticipate a series of changes in the system's behaviour. Specifically, we expect an increase in transmission delay, additional network overhead due to the tunnelling protocol, and potentially more packet loss or retransmissions depending on the quality and stability of the underlying internet connections.



```
No.   Time      Source          Destination     Protocol Length Info
10222 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5879205 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10223 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5880657 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10224 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5882109 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10225 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5883561 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10226 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5885013 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10227 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5886465 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10228 23.855977 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5887917 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10229 23.856242 192.168.1.39    104.16.230.132  TCP        54 50977 → 443 [ACK] Seq=4677 Ack=5889369 Win=3712768 Len=0
10230 23.870490 192.168.1.39    46.136.58.160   TLSv1.2   914 Application Data
10231 23.875369 192.168.1.39    46.136.58.160   TLSv1.2  1715 Application Data
10232 23.876890 192.168.1.39    46.136.58.160   TLSv1.2  1768 Application Data
10233 23.877128 192.168.1.39    46.136.58.160   TLSv1.2  1990 Application Data
10234 23.884785 192.168.1.39    46.136.58.160   TLSv1.2  3967 Application Data
10235 23.885062 192.168.1.39    46.136.58.160   TLSv1.2  2591 Application Data
10236 23.885441 192.168.1.39    46.136.58.160   TLSv1.2  1842 Application Data
10237 23.887461 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5889369 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10238 23.887461 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5890821 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
10239 23.887461 104.16.230.132  192.168.1.39    TCP      1506 443 → 50977 [ACK] Seq=5892273 Ack=4677 Win=73728 Len=1452 [TCP PDU reassembled in 10240]
```

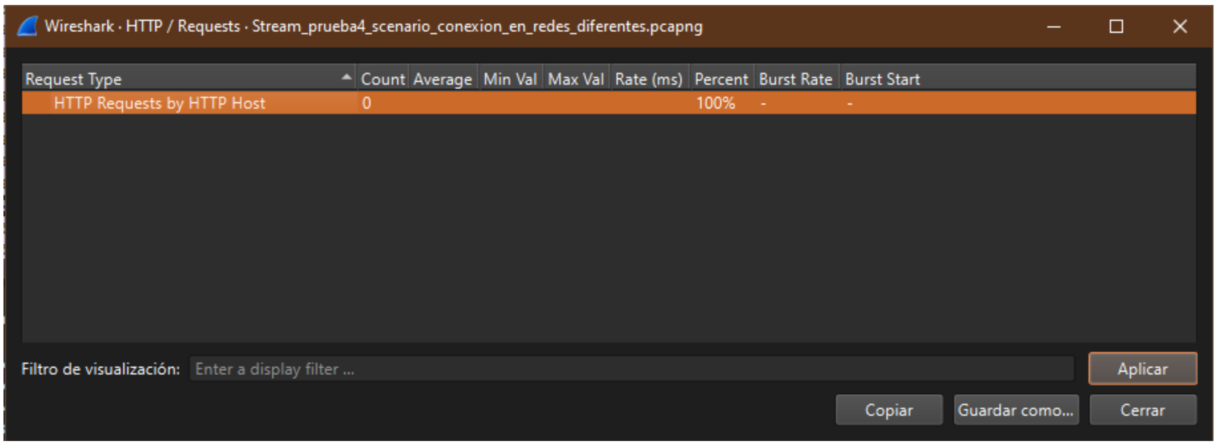
```
    Source Port: 443
    Destination Port: 50977
    [Stream index: 2]
    [Stream Packet Number: 4488]
  › [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 1452]
    Sequence Number: 5889369    (relative sequence number)
    Sequence Number (raw): 926626255
    [Next Sequence Number: 5890821    (relative sequence number)]
    Acknowledgment Number: 4677    (relative ack number)
    Acknowledgment number (raw): 2729189948
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x010 (ACK)
    Window: 9
    [Calculated window size: 73728]
    [Window size scaling factor: 8192]
    Checksum: 0x8ec1 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▾ [Timestamps]
      [Time since first frame in this TCP stream: 22.143877000 seconds]
      [Time since previous frame in this TCP stream: 0.031219000 seconds]
  › [SEQ/ACK analysis]
```

```
0020  01 27 01 bb c7 21 37 3b  31 cf a2 ac 22 3c 50 10
0030  00 09 8e c1 00 00 5e 43  39 1b 78 a8 9b c3 ad e7
0040  da 6b b8 76 ac 98 7b c6  2d 6b ae e0 c5 44 ab be
0050  01 8c 0b 14 88 3f a3 c7  00 41 59 0d 74 d1 a0 5a
0060  43 1c 02 a4 c4 8a 5e e1  89 83 bd 2f 00 c3 43 ea
0070  03 c2 3a 3a c0 78 12 fc  d7 d0 59 67 ea ba 5d 85
0080  52 af 58 e8 5d 70 34 12  86 8f 88 85 f9 7e 00 31
0090  63 22 94 ee 7c 50 09 6c  7f 52 24 ac 2e ed 32 7c
00a0  e2 a9 9d 27 d9 7c ce 6b  fc c6 b2 11 ff 4b 8e d6
00b0  da ea e0 1a 34 12 d7 e7  30 5a e9 17 69 70 45 ef
00c0  e2 e6 74 16 b7 bf 70 73  01 86 54 e1 7d 65 3a 1f
00d0  04 01 f6 e2 6b 10 10 a3  46 af 55 6f 70 19 3f a1
00e0  fb d1 01 b0 9c f8 4c c3  96 74 12 75 89 7b 0a b0
00f0  8a a3 0f 96 1b b1 dd 5f  6e fa d1 f1 48 df e4 f0
0100  8e 56 c2 13 7b 1c 1e e3  cf ea b5 cb 39 90 37 31
0110  47 b8 22 41 22 16 3e 81  7d 82 58 10 9d 8d 7f 40
0120  5e 37 a1 cb 63 87 72 30  25 b1 10 fe 10 8e d5 2f
0130  9d 17 55 70 02 70 47 00  0e fb bc f3 3a 88 60 0e
0140  57 1e ea 96 c4 6b 96 00  d5 75 6f ab 15 20 f9 64
0150  e8 85 d3 51 e4 df 5d dd  6d 5e 43 6f 65 eb 51 d7
0160  ab 0d 8a f2 14 97 8e 8d  7e d8 8c 5e b2 35 1e a7
0170  21 7f 43 31 14 1a 36 23  fb f8 8e 76 59 4a cb 1a
0180  ac c5 60 c2 c9 48 1d fc  f4 09 01 1a 65 a8 37 e8
0190  d3 aa ac 13 c5 48 3b ce  bf 57 25 7e 2a ef d7 1d
01a0  8c f9 de 6c 34 52 16 ae  20 c3 fd 55 11 51 f2 b5
```

Time relative to first frame in this TCP stream (tcp.time_relative)   Paquetes: 22176· Marcado: 1 (0.0%)   Perfil: Default

Figure 4-2 packets delay

Once the connection is established and streaming is active, we will monitor the server's statistics using the Node Media Server web interface. Given the network separation and tunnelling, it is expected that there won't be any HTTP requests and retransmissions as compared to the previous local network scenarios.



Wireshark · HTTP / Requests · Stream_prueba4_scenario_conexion_en_redes_diferentes.pcapng

Request Type	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
HTTP Requests by HTTP Host	0					100%	-	-

Filtro de visualización: Enter a display filter ...

Figure 4-3 http requests

Finally, through packet capture with Wireshark and real-time observations, we will be able to accurately measure the impact of network distance and tunnelling on stream quality, delay, and overall server performance.

The figures and analysis provided below will offer a detailed view of how the server is going on with the traffic in the scenario.

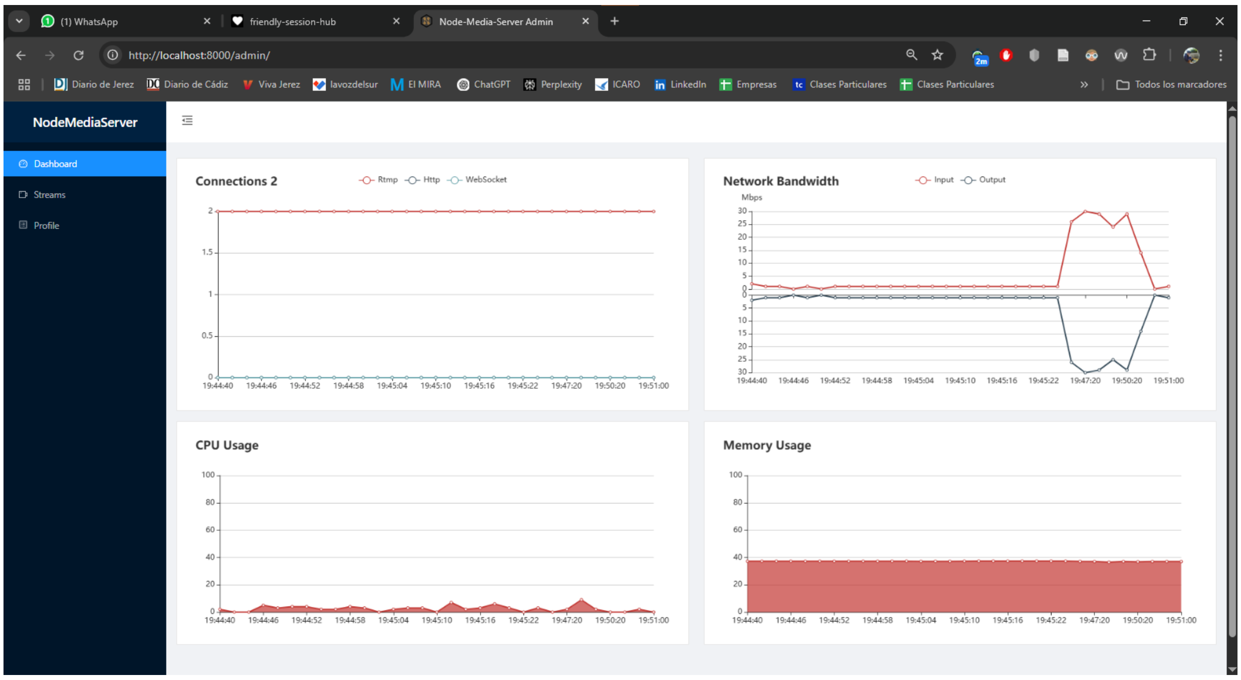


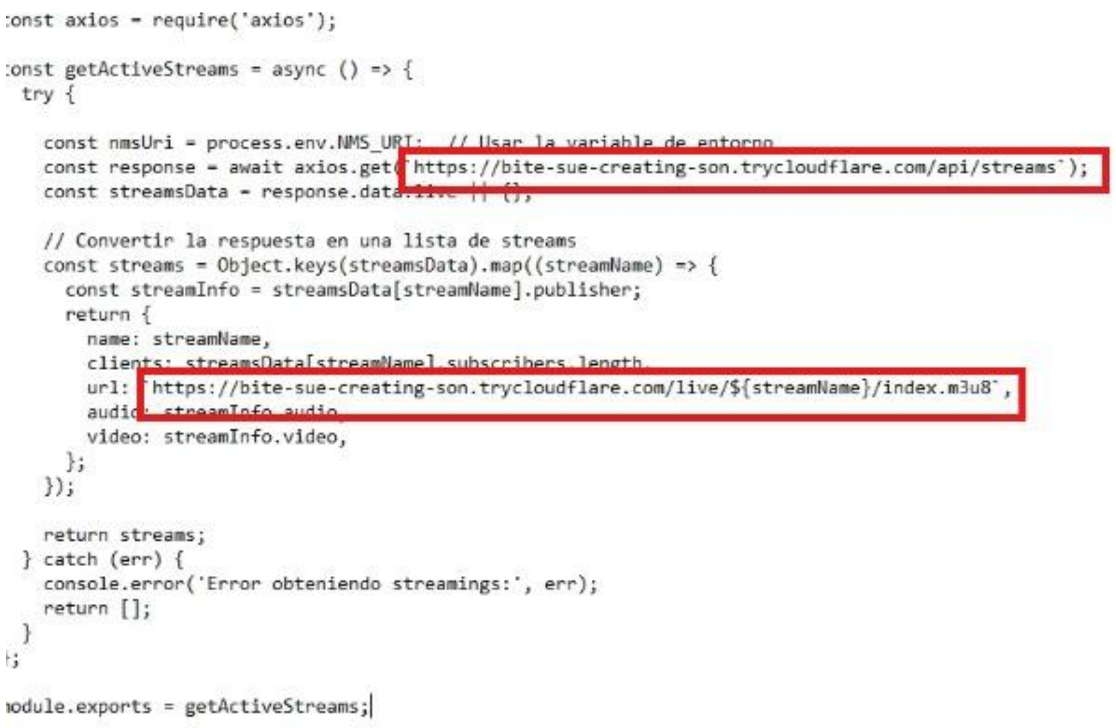
Figure 4-4 server statistics

4.1 Important changes

To make the streaming visible to users on the WAN side, it is necessary to apply a small modification to the Node Media Server's web server setup. Specifically, the code must be updated to point to the public URL provided by Cloudflare for both API calls and the playback URLs. As shown in the figure below, the URLs in the code are replaced with the public Cloudflare address:

API Endpoint Update: Instead of using the local server address, the anisette call must target the Cloudflare public endpoint

Streaming URL Update: Similarly, the stream playback URL should use the Cloudflare domain


```
onst axios = require('axios');

onst getActiveStreams = async () => {
  try {

    const nmsUri = process.env.NMS_URI;  // Usar la variable de entorno
    const response = await axios.get(`https://bite-sue-creating-son.trycloudflare.com/api/streams`);
    const streamsData = response.data.live || {};

    // Convertir la respuesta en una lista de streams
    const streams = Object.keys(streamsData).map((streamName) => {
      const streamInfo = streamsData[streamName].publisher;
      return {
        name: streamName,
        clients: streamsData[streamName].subscribers.length,
        url: `https://bite-sue-creating-son.trycloudflare.com/live/${streamName}/index.m3u8`,
        audio: streamInfo.audio,
        video: streamInfo.video,
      };
    });

    return streams;
  } catch (err) {
    console.error('Error obteniendo streamings:', err);
    return [];
  }
};

odule.exports = getActiveStreams;
```

Figure 4-5 Changes that must be applied

This modification ensures that any client on the WAN network can successfully retrieve and view the live streams hosted by the Node Media Server through the tunnel created with Cloudflare.