

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FEELT – FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA DE COMPUTAÇÃO

ALAN NICOLAS DE OLIVEIRA E SILVA
12011ECP025
LUCAS ALBINO MARTINS
12011ECP022

**REDES DE COMUNICAÇÕES II: ATIVIDADE PRÁTICA 4: TAREFA
DE PROGRAMAÇÃO STREAMING DE VÍDEO**

Sumário

1. Objetivo.....	3
2. Introdução.....	3
2.1. RTP.....	3
2.2. RTSP.....	3
3. Sobre o código.....	5
4. Implementação do código.....	7
5. Referência bibliográficas e bibliografia...	16

1. Objetivo

Tarefa de programação referente ao Capítulo de Redes Multimídia. Esse laboratório, será implementado um servidor streaming vídeo e um cliente que se comunica usando protocolo de fluxo contínuo de tempo real (RTSP) e envia os dados usando o protocolo de tempo real (RTP). Logo o objetivo é implementar o protocolo RTSP no cliente e implementar o empacotamento RTP no servidor.

2. Introdução

2.1 RTP

O Protocolo de Transporte em Tempo Real (RTP – *Real-Time Transfer Protocol*) tem por objetivo padronizar as funções utilizadas em aplicativos de transmissão de dados em tempo real, como exemplo, *streaming* de áudio e vídeo. Entretanto, o protocolo não garante a qualidade de serviço nem a reserva de recursos de endereçamento. Esse protocolo roda sobre UDP/IP, utilizando a multiplexação e *checksum* do UDP para estabelecer uma comunicação fim a fim. Os blocos de áudio ou vídeo produzidos pela aplicação são encapsulados em pacotes RTP, que são encapsulados em segmentos UDP. Ressaltando que o RTP não dispõe de mecanismos de segurança ou monitoramento da transmissão e recepção dos pacotes, dependendo, portanto, do protocolo *Real-Time Transport Control Protocol* (RTCP). O protocolo é implementado na aplicação, de forma que especifica requisitos de tempo e conteúdo da transmissão multimídia, como exemplo:

- numeração sequenciada;
- selo de temporização;
- envio de pacotes sem retransmissão;
- identificação de origem;
- identificação de conteúdo;
- sincronismo.

2.2 RTSP

O Protocolo de Fluxo Contínuo em Tempo Real (*Real-Time Streaming Protocol*) é um protocolo que permite a interação cliente-servidor entre a origem da

mídia (servidor) e usuário (transdutor). Desta forma, o usuário é capaz de ter maior controle sobre a reprodução de mídia no transdutor. As funcionalidades do RTSP são ligadas à manipulação da execução do arquivo, por exemplo:

- pausa e reinício;
- retrocesso e avanço;
- reposicionamento da reprodução.

Como o RTSP encapsula os comandos separadamente do pacote de transmissão de mídia, enviando em portas diferentes, é chamado de protocolo fora da banda. As sessões são identificadas por um identificador atribuído pelo servidor e o cliente inicia a sessão via requisição de SETUP, recebendo como resposta do servidor uma mensagem de RTSP SETUP. Assim, o cliente utiliza o identificador recebido para todas as requisições até encerrar a sessão com uma requisição do tipo TEARDOWN. Vale ressaltar que o início da transmissão é dado por uma requisição RTSP PLAY.

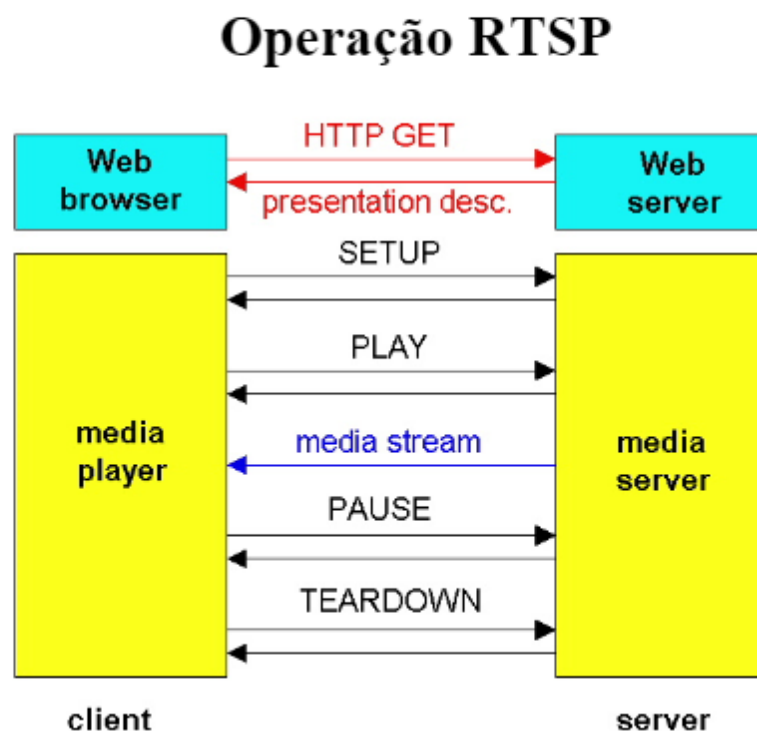


Figura 1 – Troca de mensagens RTSP.

3. Código

Seguindo a ideia do Streaming RTSP e RTP. O Trabalho de programação do livro "Computer Networking: A Top-Down Approach", de Jim Kurose, criando um código com quatro funções: Cliente, Servidor, RTPacket e o VideoStream. Logo o funcionamento do código:

- Cliente: responsável por implementar o lado cliente e a interface do usuário, utilizada para visualizar o vídeo e enviar comandos RTSP.
- Servidor: implementa o lado do servidor, responsável por responder as requisições RTSP e disponibilizar o vídeo. O servidor chama *RTPpacket* para realizar o empacotamento de dados do vídeo.
- RTPpacket: responsável por lidar com os pacotes RTP, separando cada rotina que lida com os pacotes recebidos no cliente.
- VideoStream: utilizada para ler os dados do arquivo de vídeo.

Códigos:

main_client.py

```
from PyQt5.QtWidgets import QApplication
```

```
from client.client_gui import ClientWindow
```

```
if __name__ == '__main__':
```

```
    import sys
```

```
    if len(sys.argv) < 5:
```

```
        print(f"Usage: {sys.argv[0].split('/')[0]} <file name> <host address> <host  
port> <RTP port>")
```

```
        exit(-1)
```

```

file_name, host_address, host_port, rtp_port = *sys.argv[1:],

try:
    host_port = int(host_port)
    rtp_port = int(rtp_port)
except ValueError:
    raise ValueError('port values should be integer')

app = QApplication(sys.argv)
client = ClientWindow(file_name, host_address, host_port, rtp_port)
client.resize(400, 300)
client.show()
sys.exit(app.exec_())

```

Main_server.py

```

from server.server import Server

```

```

if __name__ == '__main__':
    import sys

    if len(sys.argv) != 2:
        print(f"Usage: {sys.argv[0].split('/')[0]} <port>")
        exit(-1)
    try:
        port = int(sys.argv[1])
    except ValueError:
        raise ValueError('port value should be integer')

    while True:
        server = Server(port)
        try:
            server.setup()

```

```

server.handle_rtsp_requests()
except ConnectionError as e:
    server.server_state = server.STATE.TEARDOWN
    print(f"Connection reset: {e}")

```

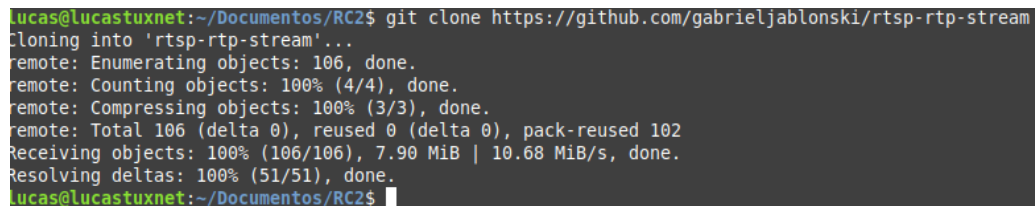
4. Implementação do código

Foi utilizado um código de RTSP e RTP baseado no exercício do livro Redes de computadores e a internet uma abordagem top-down, Kurose , Jim. Os arquivos e os códigos utilizados estão localizados no link do Github nas referências bibliograficas. O objetivo do código é implementa a funcionalidade de streaming RTSP e RTP básica usando a biblioteca Python3 padrão, além de PyQt5 e Pillow para coisas relacionadas à GUI. Mais informações disponíveis no guia de atribuições (a partir da 3ª edição, as versões mais recentes do livro podem alterar ligeiramente a atribuição). O tratamento de erros é mínimo, reabrindo o servidor e o cliente é necessário para executar outra sessão.

Para a instalação foram seguidos os seguintes passos:

1. Clonou-se o repositório *rtsp-rtp-stream* do GitHub:

Comando: git clone <https://github.com/gabrieljablonski/rtsp-rtp-stream>



```

lucas@lucastuxnet:~/Documentos/RC2$ git clone https://github.com/gabrieljablonski/rtsp-rtp-stream
Cloning into 'rtsp-rtp-stream'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 106 (delta 0), reused 0 (delta 0), pack-reused 102
Receiving objects: 100% (106/106), 7.90 MiB | 10.68 MiB/s, done.
Resolving deltas: 100% (51/51), done.
lucas@lucastuxnet:~/Documentos/RC2$

```

Figura 2 – Clone github.

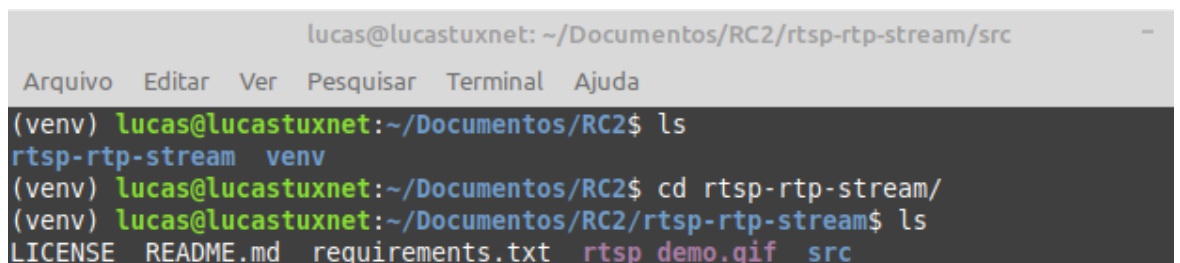
2. Com a instalação do venv (O **virtualenv** do Python é utilizado para isolar a versão do Python e das bibliotecas usadas em um determinado sistema) Criou-se um ambiente virtual no diretório *rtsp-rtp-stream*:

Comando: sudo apt install -y python3-venv

Comando: `python3 -m venv venv`

3. Ativou-se o venv (Ambiente virtual).

Comando: `source venv/bin/activate`

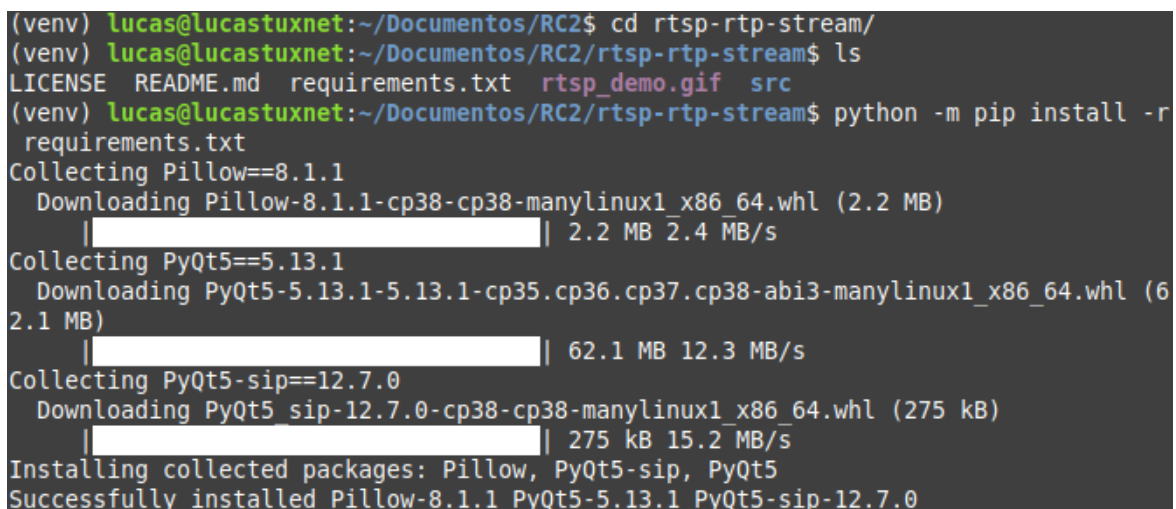


```
lucas@lucastuxnet: ~/Documentos/RC2/rtsp-rtp-stream/src
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(venv) lucas@lucastuxnet:~/Documentos/RC2$ ls
rtsp-rtp-stream  venv
(venv) lucas@lucastuxnet:~/Documentos/RC2$ cd rtsp-rtp-stream/
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream$ ls
LICENSE  README.md  requirements.txt  rtsp_demo.gif  src
```

Figura 3 – Ambiente Virtual(VENV).

4. Instalou-se os requisitos para utilizar o código Python:

Comando: `python -m pip install -r requirements.txt`



```
(venv) lucas@lucastuxnet:~/Documentos/RC2$ cd rtsp-rtp-stream/
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream$ ls
LICENSE  README.md  requirements.txt  rtsp_demo.gif  src
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream$ python -m pip install -r
requirements.txt
Collecting Pillow==8.1.1
  Downloading Pillow-8.1.1-cp38-cp38-manylinux1_x86_64.whl (2.2 MB)
    | 2.2 MB 2.4 MB/s
Collecting PyQt5==5.13.1
  Downloading PyQt5-5.13.1-5.13.1-cp35.cp36.cp37.cp38-abi3-manylinux1_x86_64.whl (6
2.1 MB)
    | 62.1 MB 12.3 MB/s
Collecting PyQt5-sip==12.7.0
  Downloading PyQt5_sip-12.7.0-cp38-cp38-manylinux1_x86_64.whl (275 kB)
    | 275 kB 15.2 MB/s
Installing collected packages: Pillow, PyQt5-sip, PyQt5
Successfully installed Pillow-8.1.1 PyQt5-5.13.1 PyQt5-sip-12.7.0
```

Figura 4 – Instalando PyQt5.

5. Para a utilização, deve-se ir para o diretório fonte dos códigos “src” dentro da pasta rtsp-rtp-stream:

Comando: `cd src`


```
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream$ cd src
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream/src$ ls
client  dummy_client.py  __init__.py  main_client.py  main_server.py  movie.mjpeg
server  utils
```

Figura 5 – RTSP-STP-STREAM.

6. Para rodar o servidor, utiliza-se a seguinte sintaxe:

Comando: `python main_server.py <port>`

O parâmetro *port* indica a porta utilizada pelo socket RTSP. Por exemplo:

Comando: `python main_server.py 5001`

7. Em relação ao cliente, deve utilizar a seguinte sintaxe:

Comando: `python main_cliente.py <file name> <host address> <host port> <RTP port>`

O parâmetro *file name* indica o nome do arquivo que será enviado via RTP, sendo o arquivo *movie.mjpeg* neste caso. Em relação aos parâmetros *host address* e *host port*, representam, sucessivamente, o endereço do servidor (*localhost* caso seja rodado localmente na máquina) e a porta selecionada para rodar o servidor. Por fim, o parâmetro *RTP port* indica a porta utilizada para receber o vídeo via RTP. Por exemplo:

Comando: `python main_client.py movie.mjpeg localhost 5001 5002`

Desta forma, utilizando os exemplos anteriores em uma implementação sobre a distribuição Linux Mint, tem-se:

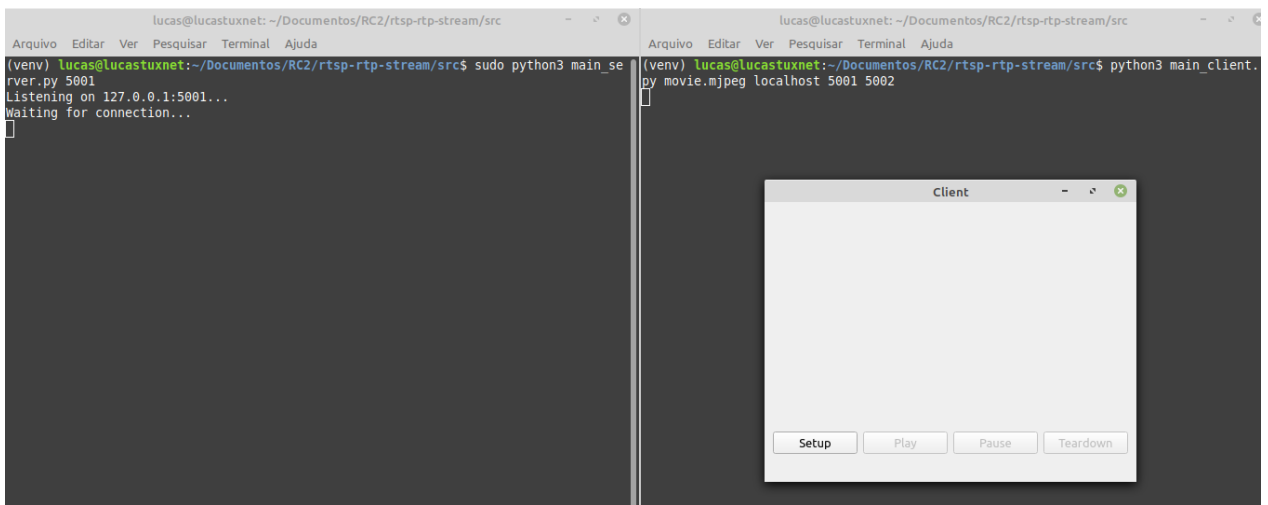


Figura 5 – Iniciando o servidor-cliente e iniciando o cliente-servidor.

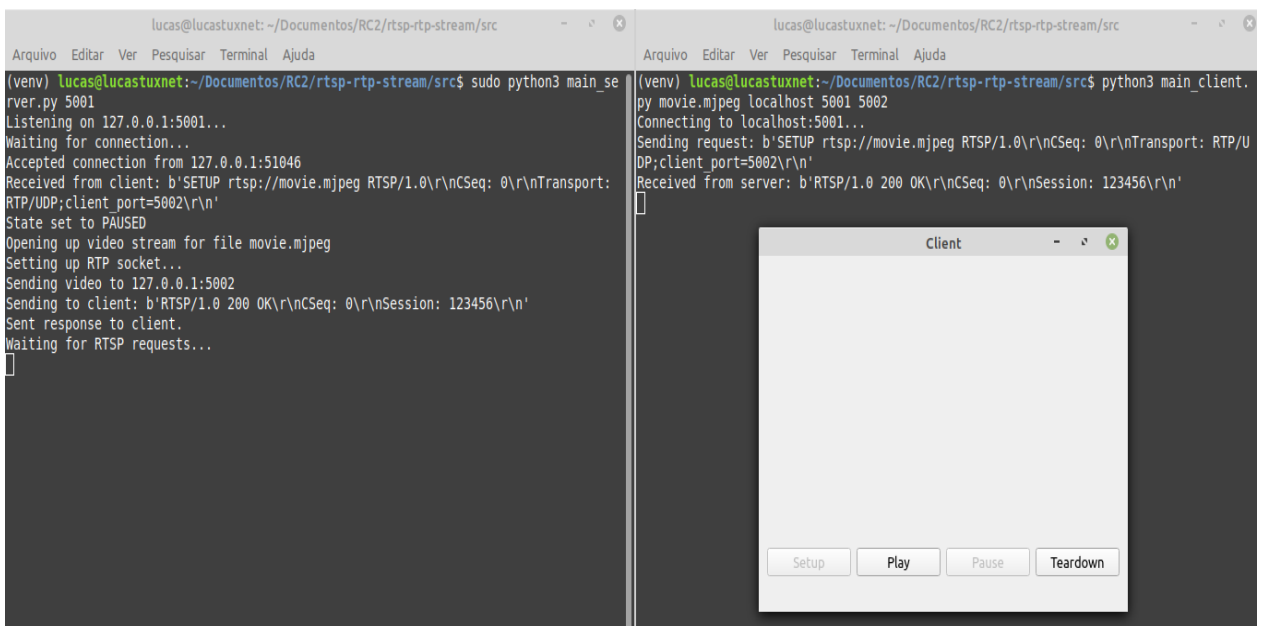


Figura 6 – Do lado direito a inicialização do servidor após o cliente iniciar o setup e do lado esquerdo a inicialização do cliente aguardando iniciar reprodução “PLAY”.

Figura 8 – Servidor executando o stream: resposta ao PLAY.

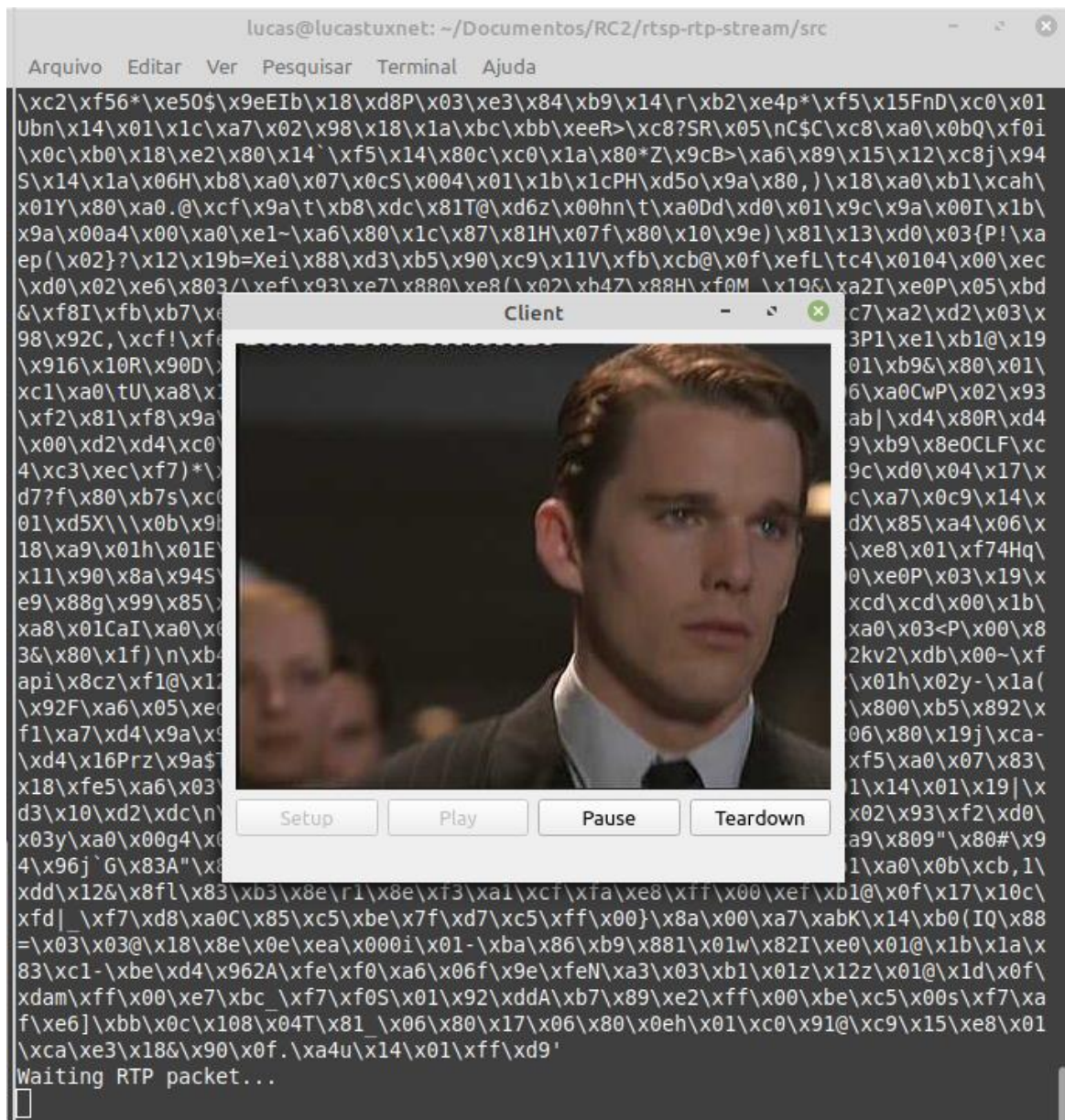


Figura 9 – Cliente assistindo ao stream: requisição de PLAY.

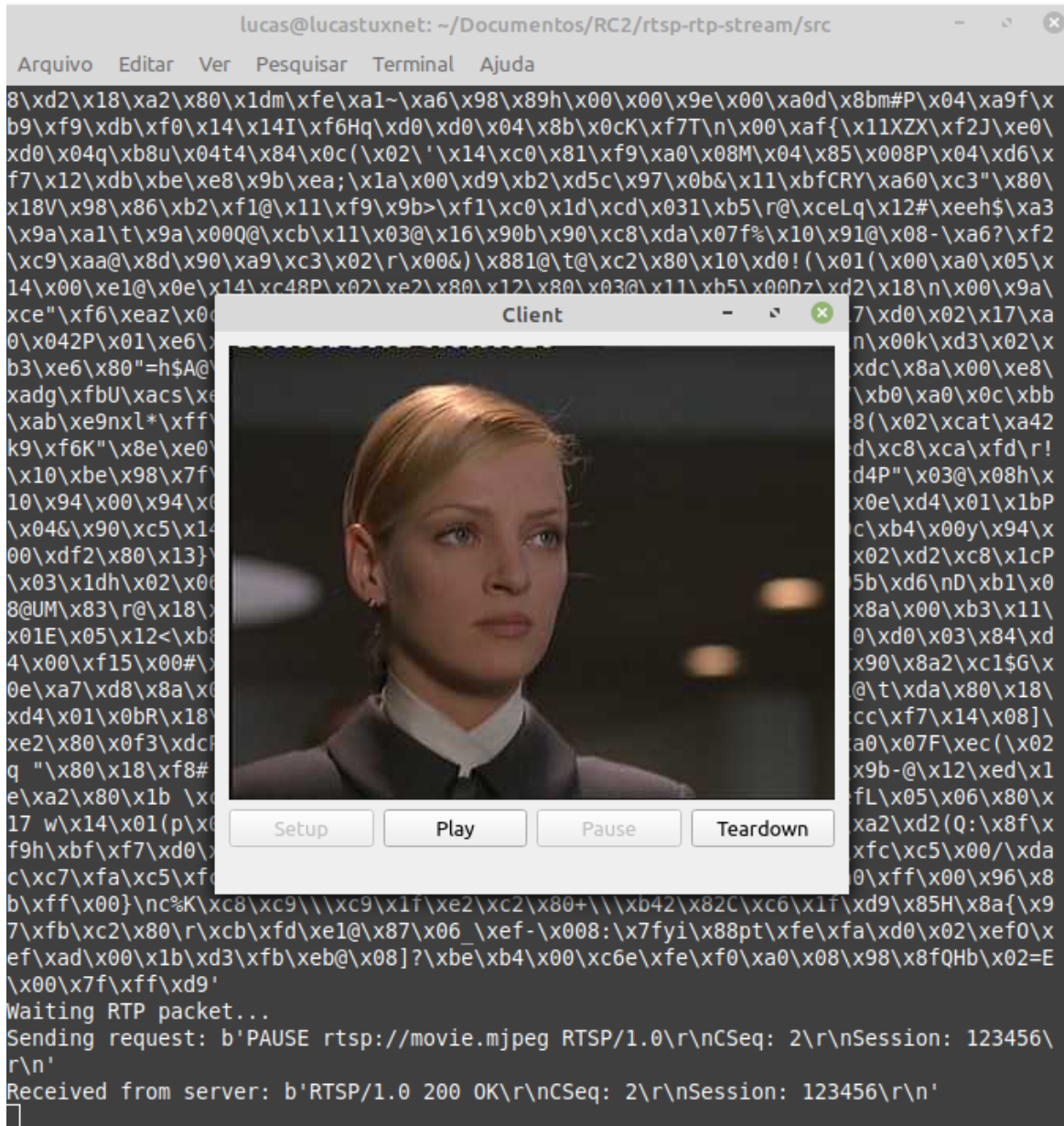


Figura 10 – Cliente ao solicitar uma pausa: requisição de PAUSE.


```
lucas@lucastuxnet: ~/Documentos/RC2/rtsp-rtp-stream/src
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

d2\x19\x16\xd2:\x8c\x11@\n\x16\x81\x0b\x0b\x80\r\xbcP\x02\x15\xe2\x80\x1aV\x80#\x8
0\x01Hc\x80\xa0\x05\x14\x86.(\x01\xb0\xff\x00\xa9\x15D\x8f\xa0\x04\xa0bP\x01@t@\x0
8i\x01\x11\xa0\x00R\x18\xe1AC\xc5\x009P\xb1\xc2\x820\xb0\xa0t\xd2\xc6\xe5\xcf\x11\
xe0z\xb1\xa0e\xa8\xf4\xc91\xfb\xc9\x14P\x03\xce\x94\xa7\xfe[\x1ab\x154\xb5\x07\x99\
x89\x14\x00\xf9\xa0Kh\x83\xa7@\xc3$\xd2\x10\x80\xe4U\x00\xc6\xebHd\x12\x1cPI^Jb*\xb
fZ\x00m\x00(\xa0\tm\xe7{w\xdc\x98*z\xa9\xe8i\x0c\xde\xb2\x9e\xde\xe9s\x1f\x0c:\xa1\
xea(\x02\xe8\x89i\x00m\xc5\x00!\xa0cA\xc7\x07\x95\xa0\x07\x16\n\xa4\x92\x02\xeI\xa
6#\xf6\x00T\xdf\x98\xed\x89\x03\xbb\xf74\x01\x93L\x91M\x00\x14\xc0Q@\x16#<R\x193'
\x987\x01\xf3\x8e\xa2\x80"\x14\x081@\x05\x00\x18\xa0\x060\xa0\x08\x98P\x02\nC\x1c\x
05\x00;\x14\x86\x14\x00\xc8?\xd4\x8a\xa2G\xd0\x02P0\xa0\x04\xa0\x02\x80\x11\xa9\x0c
\x88\x8c\xd0\x05\x98l'\x94}\xd0\x83\xd5\xa8\x19r=*0?y)>\xca(\x02\xd4v\xb6\xb1\xf4\
x89I\xf5jc&VU\x18\n\x004\x00y\x9c\xd0\x02y\x94\x00\x9eo\x14\x00\x9eo4\x00\xc9H\x9e\
x17\x88\x9f\xbc?#\xd8\xd0\x05;i\xcb\xa6\x1b\x87\xe8G\xb8\xa0t\xc9\xcd \x90\xf1@\x
15\\\x93L\x82\x06\xa0\x06\xe2\x81\x05\x00(\xa0\x05\x04\xab\x06RU\x87B\x0e\x08\xa0\r
KMjX\xb0\xb7+\xe6\xaf\xf7\x87\x0c)\x0c\xd8\xb7\xb8\x86\xea-\xf0\xb8':\x8e\x850\xb8\
xa9(s\xe1T\xb3\x10\x00\xeaMl\x14&\xd4\xadb\xc8\xff\x00X\xdf\xec\n\x00\xca\xbb\xbe\x
92\xe4\xe0|\xb1\xf6Pj\x89*\xf5\xa0\x02\x98\x05\x00\x1d\xa8\x01\xea(\x02\xc4#"x91E\
x85R(\x02'\x07v@\xe0\x9a\x04\x01X\x8f\xbah\x01\xb4\x08;P\x03\x1a\x81\x915\x00 \xa4
\x03\x80\xa0\x07\x01Hb\xd0\x04p\x11\xe4\x8c\x9095D\x8e\xca\xfa\xa8\x002=E\x03\x13#\
xd4P\x01\x91\xea(\x00\x18$\x00W\xf14\x011\xb5]\x9970\xfd\x03R\x19f\x01k\x00\x05\x1e
=\xc3\xa9,(\x02cs\x17\xfc\xf50\xfb\xe8U\x0ca\xbb\x8f?\xeb\x17\xfe\xfa\x14\x00\xdf\x
b5'i\x16\x90\x83\xedK\xfd\xf5\xff\x00\xbe\x85\x00'\xda5\x83\xe6/\xe6(\x00\xfbB\x7
f\xcfE\xff\x00\xbe\x85\x00'\xda\x13\xfb\xeb@r7\n:.\xd0\x02y\xe9\xfd\xf5\xa0nr?\x
97r]\x0eU\xb94\x01n;\x85#\x0c\xc0n@+\xc8\x8d\xff\x00-\x17\xf3\x14\x01Y\xca\x9f\xe2
\x14\xc4Dq\x8e\x08\xa0\x06b\x81\x05\x00\x14\x00\xb4\x00\x86\x98\x8d}\x0e\xe2\x0b{Y\
x04\xb2\xaa;\xc9\x93\xf4\x02\xa4\xa2\xae\xb99\x96\xf4yr\x07\x89P}\xd3\x90)\x88\xa2\
x0eE\x00:\x80\n\x00)\x80P\x02\xd0\x03\xc6=E\x00Y\x81\xd4/, \xa3\xf1\x02\x91D\x8d4'}\
xf54\x016\x9bt\x91\xc7&\xe7E\xcbt$P2\xf7\xdb\xa2#\xfd|c\xfe\x06(\x01$\x92\xcac\xfb\
xd9 '\xfd\xf0r\x02*\xc9kh\x7f\xd5\xdeF\xbe\xcc\xe0\xd0\x05I\xad\xc2\x0c\xac\xf6\x
ee?\xd9\x90P"\xa3c=E\x00 #\xd4R\x01\xc0\x8fQ@\x0e\x05\x7f\xbc)\x0c\x0b\x7f\x85\x00
\x7f\xff\xd9'
Waiting RTP packet...
Sending request: b'PAUSE rtsp://movie.mjpeg RTSP/1.0\r\nCSeq: 4\r\nSession: 123456\r\n'
Received from server: b'RTSP/1.0 200 OK\r\nCSeq: 4\r\nSession: 123456\r\n'
Sending request: b'TEARDOWN rtsp://movie.mjpeg RTSP/1.0\r\nCSeq: 5\r\nSession: 1234
56\r\n'
Received from server: b'RTSP/1.0 200 OK\r\nCSeq: 5\r\nSession: 123456\r\n'
(venv) lucas@lucastuxnet:~/Documentos/RC2/rtsp-rtp-stream/src$
```

Figura 12 – Cliente encerrando a transmissão: requisição de TEARDOWN.

