Asian Journal of Advances in Research

5(1): 860-867, 2022

GETTING RTSP TO WORK NATIVELY IN THE BROWSER

ASHLIN DARIUS GOVINDASAMY [a*]

[a] Department of Computer Science, University of South Africa, South Africa.

AUTHOR'S CONTRIBUTION

The sole author designed, analyzed, interpreted and prepared the manuscript.

Received: 09 March 2022
Accepted: 19 May 2022
Published: 01 June 2022

Original Research Article

ABSTRACT

Real-Time Streaming Protocol (RTSP) is an application-level network communication system that transfers real-time data from multimedia to an endpoint device by communicating directly with the server streaming the data. The Real-Time Streaming Protocol (RTSP) is tried-and-true video technology. It's used to control audio/video transmission between two endpoints and facilitate the transportation of low latency streaming content across the internet. Real-Time Streaming Protocol (RTSP) allows you to pull a live video stream from your camera and view it from different devices and programs. Its primary uses are to pull a video feed from a camera to an NVR, viewing software, or even home automation solutions. RTSP is not natively supported in Web Browsers at the time the paper was written.

In this paper, I will discuss techniques on how to render RTSP natively in your web browser using only a *<img>* tag inHTML (Hyper Text Markup Language) powered by my platform which hundreds of people use today called OpenRTSP, and an alternative way to build an engine/server like my platform from scratch in detail.

Keywords: RTSP; web; streaming; protocol; HTML; CCTV; camera; image; processing.

1. INTRODUCTION

RTSP is a real time streaming protocol which is actually the pure video or audio feed which comes off IP cameras or networked DVRs. It is worthy if you are facing the issue of frustration with camera that locks you in their devices (Techwin & Vision, n.d.). Real time streaming protocol is actually application layer protocol which is used for directing streaming media servers with pause and play features. This system provides real time control of streaming media [1]. First RTSP was invented by Real Networks, Netscape and Columbia University [2,3]. The first draft of RTSP was submitted to IETF on October 1996 by two companies namely Progressive Networks & Netscape [4,5]. Further drafts which includes merged drafts were submitted by the control working group MMUSIC WG (*Https://Datatracker. Ietf.Org/Doc/Html/Draft-Ietf-Mmusic-Rtsp-01.Html 1/59*, 1997) [6]. The authentic protocols for real time streaming was published in 1998 (Https://www.stal.pt/phocadownload/2020/200528_Resoluo%20suplemento%20de%20penosidade%20e%20risco.pdf, 1998) [7].

What does aReal-Time Streaming Protocol (RTSP) do and look like?From what we know from looking at the abstract. Basically, it is a protocol for allowing us to stream video/audio footage from various sources. It encodes and compresses the stream into a lightweight network packet for us to see in a desktop application or web portal (decoded by a server) [8].

We can view an RTSP stream by entering the URL of the stream in an RTSP streaming software.

**Corresponding author: Email: javeriaumber474@gmail.com;*

Example URL: rtsp://admin:12345@192.168.1.210:554/Streaming/Channels/101, where the admin is the username and 12345 is the password.

All security cameras support the RTSP video stream, which means the end-user can use media players such as VLC to watch the live stream remotely from anywhere. Real-Time Streaming Protocol (RTSP) is a protocol that is used to transfer real-time audio or video between a client and a server. The protocol is used for establishing and controlling media sessions between endpoints [9].

Why did I start to investigate RTSP streams? As I moved from a beach flat in Durban, South Africa to a huge house in Somerset West. I have many big home projects I want to implement as time progresses. As I am extremely security conscious, I wanted to make my own app to track all the cameras at home regardless of what camera brand I have. The current issue with these applications is that I must have specific camera brands linked to the brand app. There is no generic app to do that. When I investigated rendering RTSP streams on the web browser. It was a pain to do it. I either must have FFMPEG installed or use painful libraries to do so. I ended up building OpenRTSP.com to make life easier for developers by just using a <img> tag to render the RTSP stream piggybacking off my Open RTSP.com server for free. The developers can also deploy OpenRTSP.com locally and modify it to their liking. What I happened to accomplish to solve my problems above was by making an easy-to-use platform to render RTSP streams for free natively in the browser.

2. AIM OF THIS STUDY

Till now there was a limitation in getting an RTSP stream to render natively in a web browser, you either must have a client-server for your applications or install tons of libraries to make it work. This study concluded by building a platform for rendering RTSP streams in an <img> HTML tag for developers. The author made it simple enough to just take the RTSP stream link convert it to base16 and merge the link to the OpenRTSP.com API system then get a streamable link on OpenRTSP.com and set the *src* of a <*img*>tag to render that RTSP stream natively in any given browser.

This study also shows any developer in detail how to replicate the construction of this platform and techniques discovered by the author.

3. METHODOLOGY

3.1 Examples of Streaming RTSP

I will show you examples of streaming RTSP. Using Big Bunny Stream as a test.

The RTSP link we will use for testing is: rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mp4

Note OpenRTSP.com is the platform We built. We will discuss more OpenRSTP.com in detail in later parts of the paper.

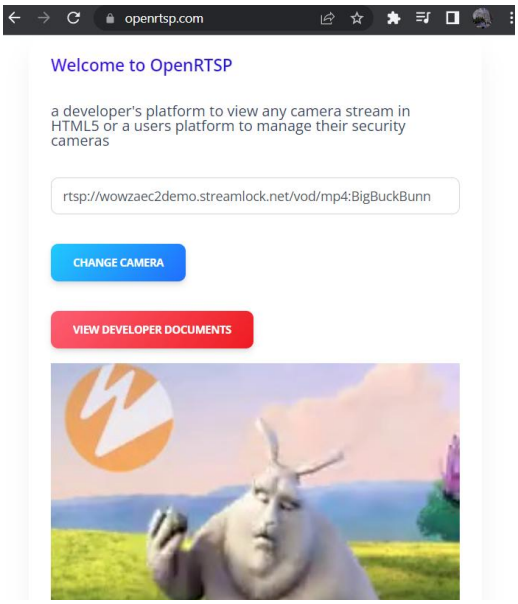

Fig. 1. Using my platform OpenRTSP to render the stream in the native browser using the <img> HTML tag

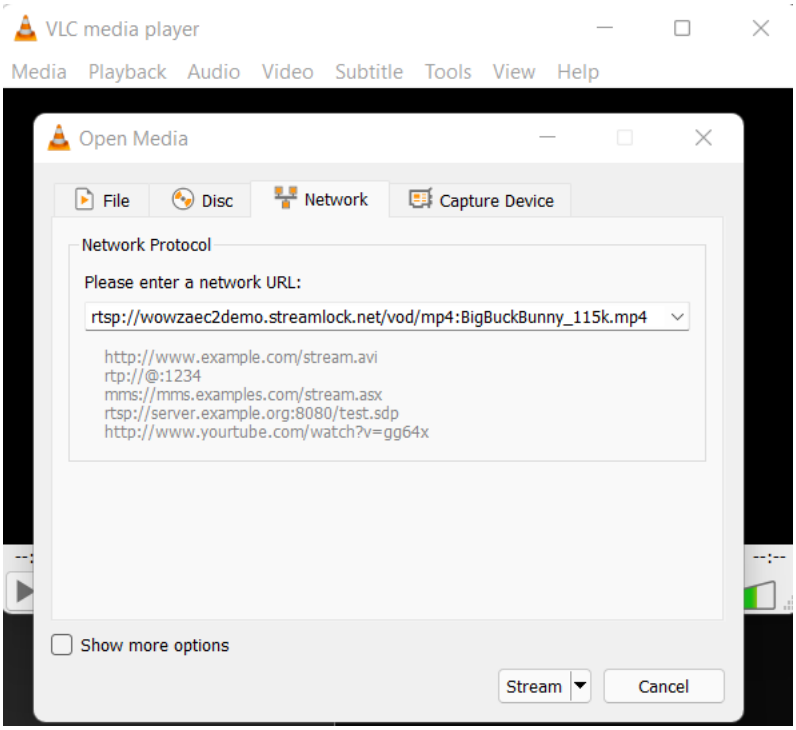

Fig. 2. Using VLC to render the RTSP link

VLC is a software desktop application used to open videos with fancy CODEXs.

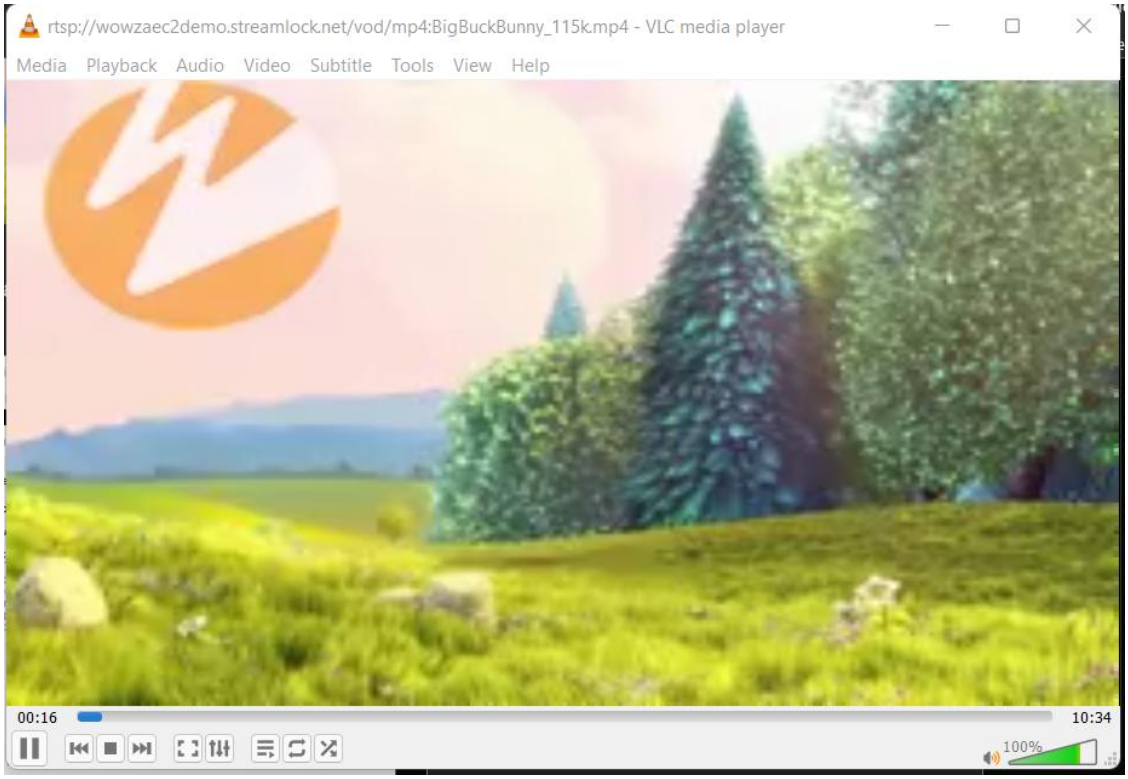

Fig. 3. VLC Video/audio output of the big bunny stream

3.2 Creating the Web RTSP Streaming Platform

3.2.1 Approachs to render a RTSP streams for a web browser

My approach will use image processing to process the stream into a browser readable format. Create a server that will render the RTSP stream using my code and libraries for *Nth* RTSP streams.

Basically, in summary, the RTSP devices will go through the filter (in this case the server) to push out a moving image stream on the web-server route for the clients to open in the web browser.

My goal wasto build a platform to cater to the above. I also wanted to make it easy for developers to just type in a few lines of code, also use zero libraries, have no server to piggyback on, and just a native HTML tag for them to use. I believe I succeeded in accomplishing my goals.

If you look at the diagram below this is the summary of what I wrote above, on how it is going to work out.

3.2.2 Tools and libraries needed to build the RTSP conversion server

We need a web server hosting site– We will use Heroku.com which uses AWS to run our container if you want the World Wide Web to use your rendering server else you just can run the webserver localhost. Running your server localhost can be port forwarded on your network to the world wide web so others can also use it.

A library that can do Image Processing for us in our case we used OpenCV a python library.

A web framework. We used a stack powered by the ADGWEBSDK which is created by Python, Vanilla JavaScript or React.js or Angular, etc. ADGWEBSDK uses the Flask Framework which also has Django on the stack.
Flask handles all the web stuff such as routing and HTTP Traffic.

3.2.3 Using opencv to view RTSP stream on python locally

When starting a software engineering problem, I normally tend to get the core functionality sorted out first. In this paper, my goal was to render the RTSP stream natively on the browser. First, I started off by Image Processing the RTSP stream into a human viewable format.

The code snippet above displays the stream into a window frame of the Big Bunny RTSP stream.

As you can see the cv2 module in python does the heavy lifting and Image Processing for us. It is making life easier for us. The core functionality is done. What we need to do next is just make it work in a web service/server environment. This is where we use Flask and the ADGWEBSDK Stack.

To convert our general code into HTTP route code we are just going to return each Open CV frame as an image object in the Flask Framework. If you look below this is how I did it for Open CV and you can also do it like that.

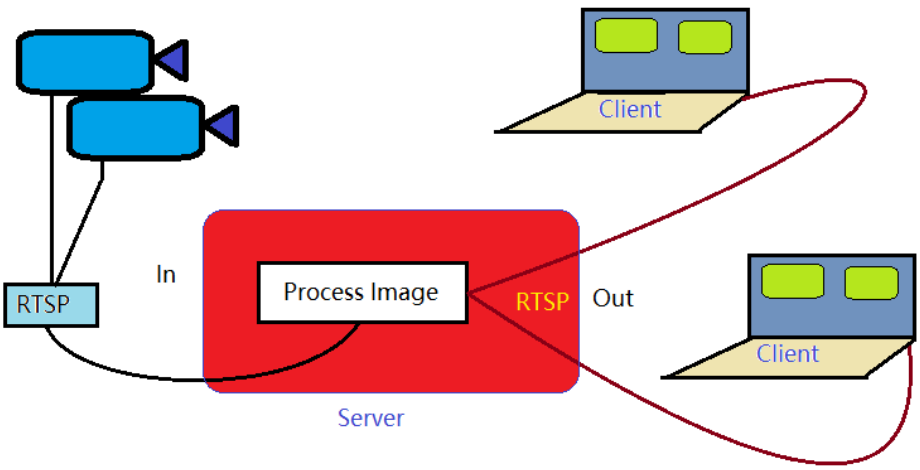

Fig. 4. Hierarchy diagram of the system

```python
import cv2
cap = cv2.VideoCapture(
    "rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mp4")
if not cap.isOpened():
    print('Cannot open RTSP stream')
    exit(-1)
while True:
    _, frame = cap.read()
    cv2.imshow('RTSP stream', frame)
    if cv2.waitKey(1) == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Fig. 5. Code snippet of python module Open CV Image Processing the Big Bunny RTSP stream into a viewable format

```python
@app.route('/video_feed/<rtsplink>')
def video_feed(rtsplink):
    try:
      rtsplink =  base64.b16decode(rtsplink).decode('UTF-8')
      print(rtsplink)
      camera = cv2.VideoCapture(rtsplink)
      return Response(openrtsp.gen_frames(camera), mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
      print(e)
      return send_file('./static/img/novideo.png', mimetype='image/gif')
```

Fig. 6. Code snippet ofthe GET REQUEST web route, for displaying an RTSP video feed on the browser as a <img> tag

As you can see above, we are using an HTTP route of */video_feed/<rtsplink>*
<rtsplink> is a variable that should be in base16 format. We recycled our first method code for testing rendering a basic RTSP stream on OpenCV on our desktop environment. The only difference here is that it now can take an Nth number of streams and render it concurrently on the server.

That's basically about rendering RTSPs streams on the web browser. OpenCV does the hard work for us. OpenCV is not restricted to only Python. OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License. It was originally written in C and C++.

All I did in this paper is show how easy it is to render these RTSP streams on the web browser and give your guys a general perspective on what you should do when tackling a problem with rendering RTSP feeds. My platform OpenRTSP works like this and any one of you can use it free of cost.

As the paper progress, I will show you how to use OpenRTSP, and give you my code repository to play around with.

3.3 Some Real-World Use Cases Of Open RTSP and this Paper Approach To Rendering RTSP Streams

- You can build a cool CCTV camera application
- Build AI Camera Software with Motion Detection etc.
- OpenRTSP code is open source you can contribute to the project or just bootstrap the code and privately use it for your personal project.
- Many more creative ideas you can think of.

3.4 Using OpenRTSP to Render a Stream on Your Web Page

3.4.1 Java script function to convert the link to base16

Firstly, you need a function to convert a given string into base16. We will use the function to convert our RTSP URL into base16 so our web server OpenRTSP can read the RTSP URL in the URL Route and render the RTSP stream into a feed our browser can understand.

3.4.2 Getting the stream to work using OpenRTSP

How do we get the GET Request?

According to the OpenRTSP.com API Route, we will be using below.
/video_feed/<rtsplink> this route requires the var - as base16
For example, using the Big Bunny Stream.
we just convert this camera string to base16 which is from
rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mp4
to
727473703A2F2F776F777A6165633264656D6F2E73747265616D6C6F636B2E6E65742F766F642F6D70343A4269674275636B42756E6E795F3131356B2E6D7034
Now append the base16 to the GET request which equates
https://openrtsp.com/video_feed/727473703A2F2F776F777A6165633264656D6F2E73747265616D6C6F636B2E6E65742F766F642F6D70343A4269674275636B42756E6E795F3131356B2E6D7034

```
function toHex(str) {
    var  result  =  '';
    for (var  i=0; i<str.length; i++) {
    result  +=  str.charCodeAt(i).toString(16);
    }
    return  result.toUpperCase();
}
```

Fig. 7. JavaScript Function to convert string to base16

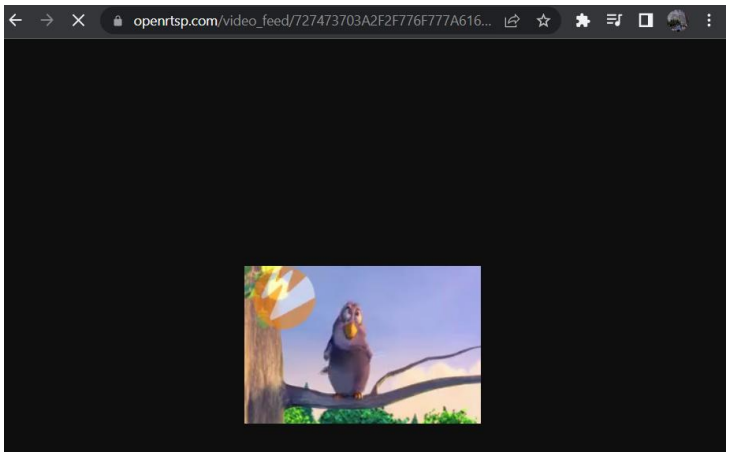

Fig. 8. Our RTSP Stream rendered in pure HTML as a <*img*> tag

To use it on your web application just create a <img> tag and set the source attribute to the openrtsp.com link you generated. In Big Bunny Case. Your HTML tag should be.

<img src="https://openrtsp.com/video_feed/727473703A2F2F776F777A6165633264656D6F2E73747265616D6C6F636B2E6E65742F766F642F6D70343A4269674275636B42756E6E795F3131356B2E6D7034">

3.5 Github Repo of Openrtsp

On my GitHub repo you can find all code for the website you see on OpenRTSP.com

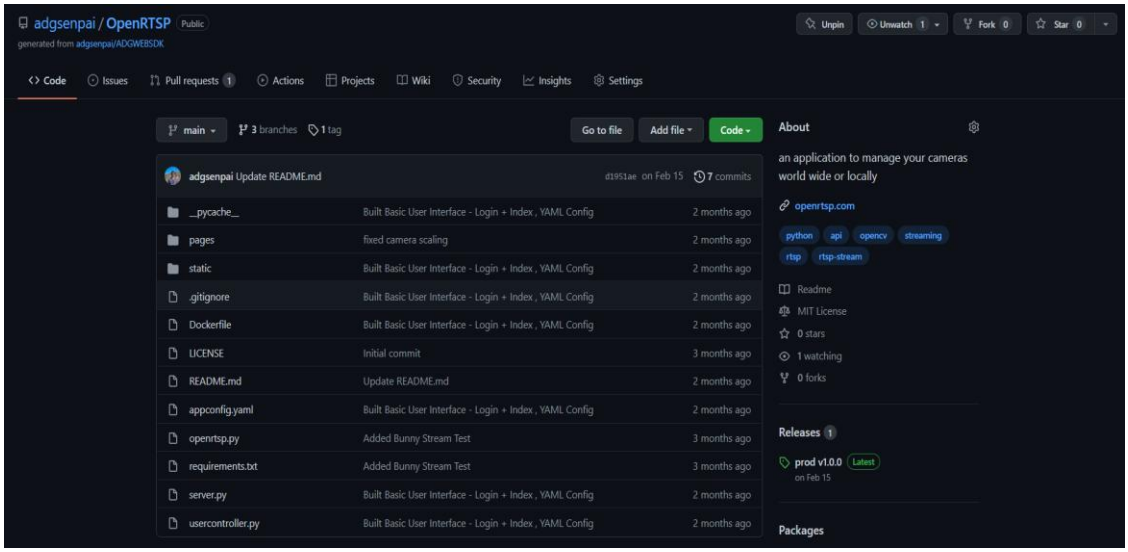

Fig. 9. GitHub Repo of OpenRTSP.com

3.6 Developer Documents

You can view the official developer documents for my platform using the URL.
docs.openrtsp.com

In the documentation, you can find out how the app works in detail, how to use it.

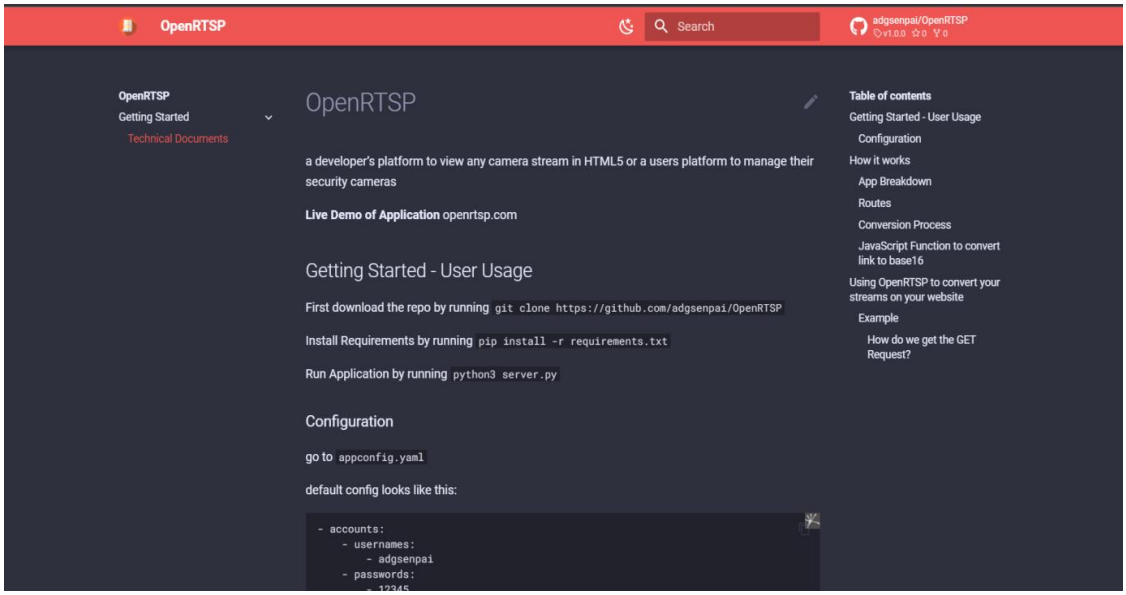

Fig. 10 Developer Documentation of OpenRTSP.com

3.7 What Does Openrtsp Do?

The platform serves many purposes:

- It is an Image Processing conversion server for RTSP streams that get converted into a Web Response.
- It can manage your cameras to the *Nth* Degree number of cameras.
- There aremanymore creative opportunities you can do with this platform which I won't mention.
- You can run it as your private RTSP decoder server for more performance and latency or use the public OpenRTSP.com server.

4. CONCLUSION

This paper explains how RTSP stream work, the Image Processing of RTSP, Creation of a webserver to render the RTSP stream into web browser streamable format. How to use the platform created in this paper (OpenRTSP) to render the RTSP stream using just native HTML code.

5. LIMITATIONS

The only limitation with this approach is that the sound cannot be rendered using a <img> tag else everything works brilliantly.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Ruether T. RTSP: The Real-Time Streaming Protocol Explained | Wowza. Wowza Media Systems, May 6. 2021;1–6. Available:Https://www.wowza.com/blog/rtsp-the-real-time -streaming-protocol-explained
2. Worldcat S, Mateo S, Physical A, Resource I, File C. InfoWorld: the newspaper for the microcomputing community. 1980;1–2.
3. BAZZAZ F. No eBook available. Food and Agriculture Organization of the United Nations, 69.
4. david l glaser. (1997). No Covariance structure analysis of health-related indicators in the elderly at home with a focus on subjective health Title. Icassp. 2020;21(3):295–316.
5. Lanphier R, Schulzrinne H, Schulzrinne H, Francisco S, Schulzrinne H, Jose S, Schulzrinne H. Real-Time Streaming Protocol (RTSP) Port Numbers. 1997;5–6.
6. Rfc AM, Time R, Protocol S; 1998. Available:Https://datatracker.ietf.org/doc/html/draft-ietf-mmusic-rtsp-08.html 1/92. C.
7. Vladimir Dr., VF. No Title No Title No Title. Gastronomía Ecuatoriana y Turismo Local. 1967;1(69):5–24. Available:https://datatracker.ietf.org/doc/html/draft-ietf-mmusic-rtsp-01.html 1/59. (1997). February, 1–59.
8. Santos H, Cruz RS, Nunes MS. Rate adaptation techniques for WebTV. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 40 LNICST. 2010;161–168. Available:Https: // doi.org/10.1007/978-3-642-12630-7_19
9. Options V, Options AV, Options A, Audio A, Evaluation E, Decoders V, Decoders A, Encoders A, Options MC, Levels D, Information AP, Options OM, Information EB, Ac-, O., Options, E., & Encoders, V. ffmpeg Documentation Table of Contents. 2012;1–119.