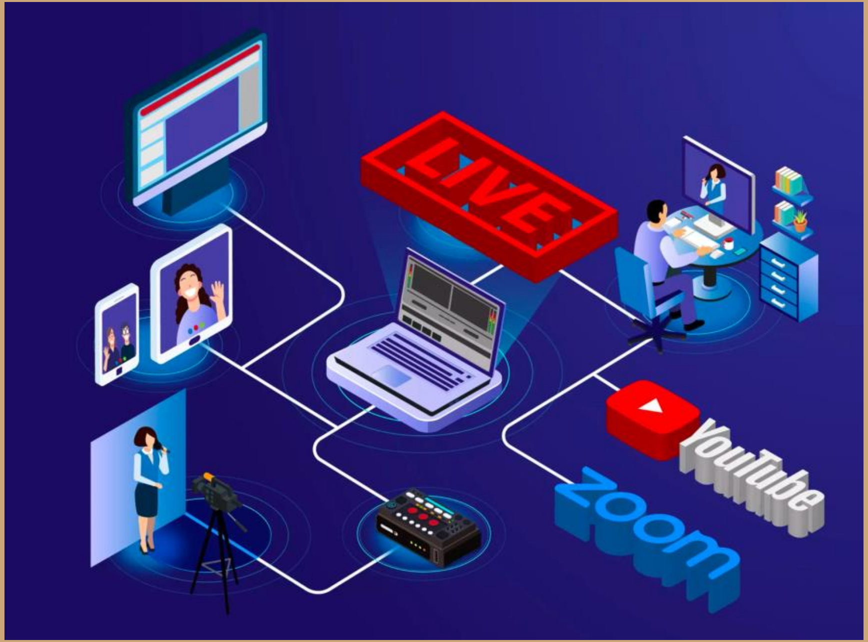
StreamVerse: A Comprehensive Multi-Client Streaming Solution for Academic Environments



Project Instructor - Prof. Anil Kumar Singh
Department of CSE
MNNIT Allahabad

Team Members (Group No. - 21)

- Nishchay Chaurasia (20204129)
- Niraj Kumar Gupta (20204128)
- Nilesh Malav (20204127)
- Nitesh Rana (20204130)

Contents

Introduction

- Multi-client streaming is a type of multimedia streaming that allows multiple users to access and view the same media content simultaneously.
- Multi-client streaming is widely used in applications such as live video streaming, web conferencing, and online training platforms.
- Multi-client streaming eliminates the limitations of the traditional client-server model, allowing for seamless streaming to multiple clients without overloading the server.
- The system enables the efficient delivery of high-quality streaming content to a large number of clients with low latency and fast response times.

Motivation

- ## High-quality Streaming

  Multi-client streaming is a rapidly evolving technology that enables the efficient delivery of high-quality streaming content to a large number of clients with low latency and fast response times.

- ## Traditional client-server model

  The traditional client-server model has limitations in handling multiple client connections simultaneously, and multi-client streaming addresses these limitations by using various technologies and protocols.

- ## Multiple video feeds Simultaneously

  Multi-client streaming also enables the ability to view multiple video feeds simultaneously, which is particularly useful for live event coverage.
  With multi-client streaming, viewers can choose which event they want to watch, and the streaming server can deliver multiple live video feeds simultaneously.

Motivation

- Cross Platforms

  Multi-client streaming gives the flexibility in terms of the devices and platforms that can access the stream, making it accessible to a wide range of users.

- Reduce Individual bandwidth consumption

  Multi-client streaming allows for scalable and efficient distribution of video content to a large number of users, without the need for individual streaming connections for each user.

- Enhanced Feature

  It allows for real-time feedback and interaction between viewers, which can facilitate collaboration and knowledge sharing and can be useful and different fields.

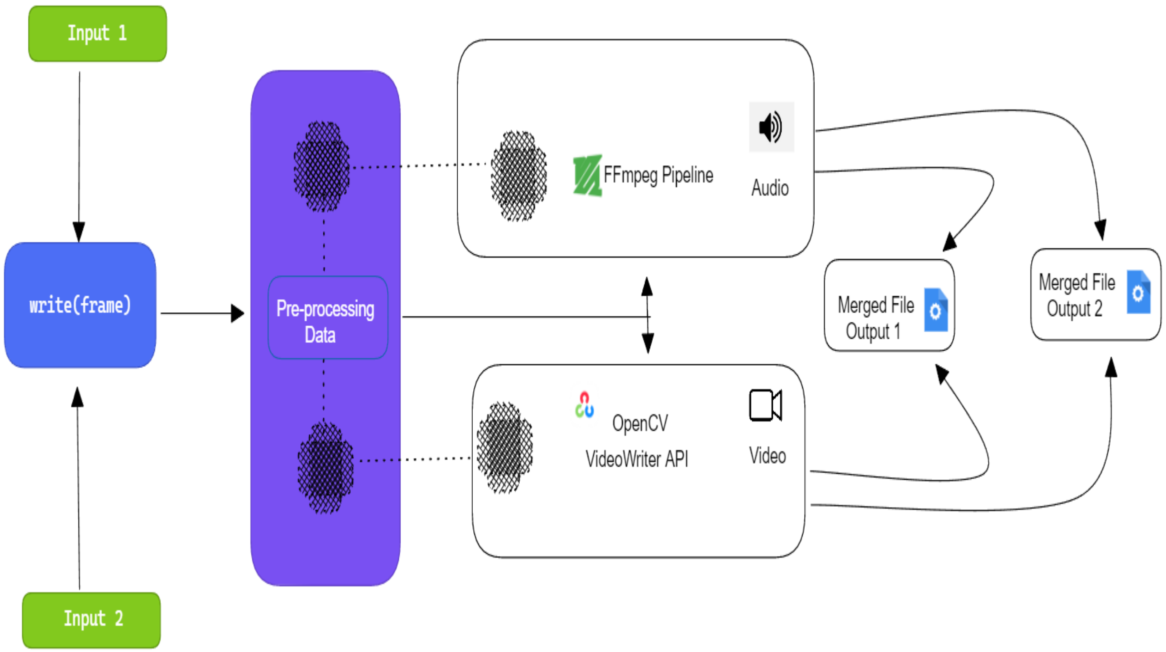
Architecture and Design

- **Client devices**: These are the devices that connect to the streaming server to receive the video feed. They could be smartphones, laptops, or other internet-enabled devices.
- **Streaming server**: The server that sends the video stream to the clients. It receives the video input from a camera or other sources, processes it, and sends it out to the clients.
- **OpenCV and FFmpeg**: These are libraries used for image and audio processing, respectively. They are used to capture and process the video stream from the source.

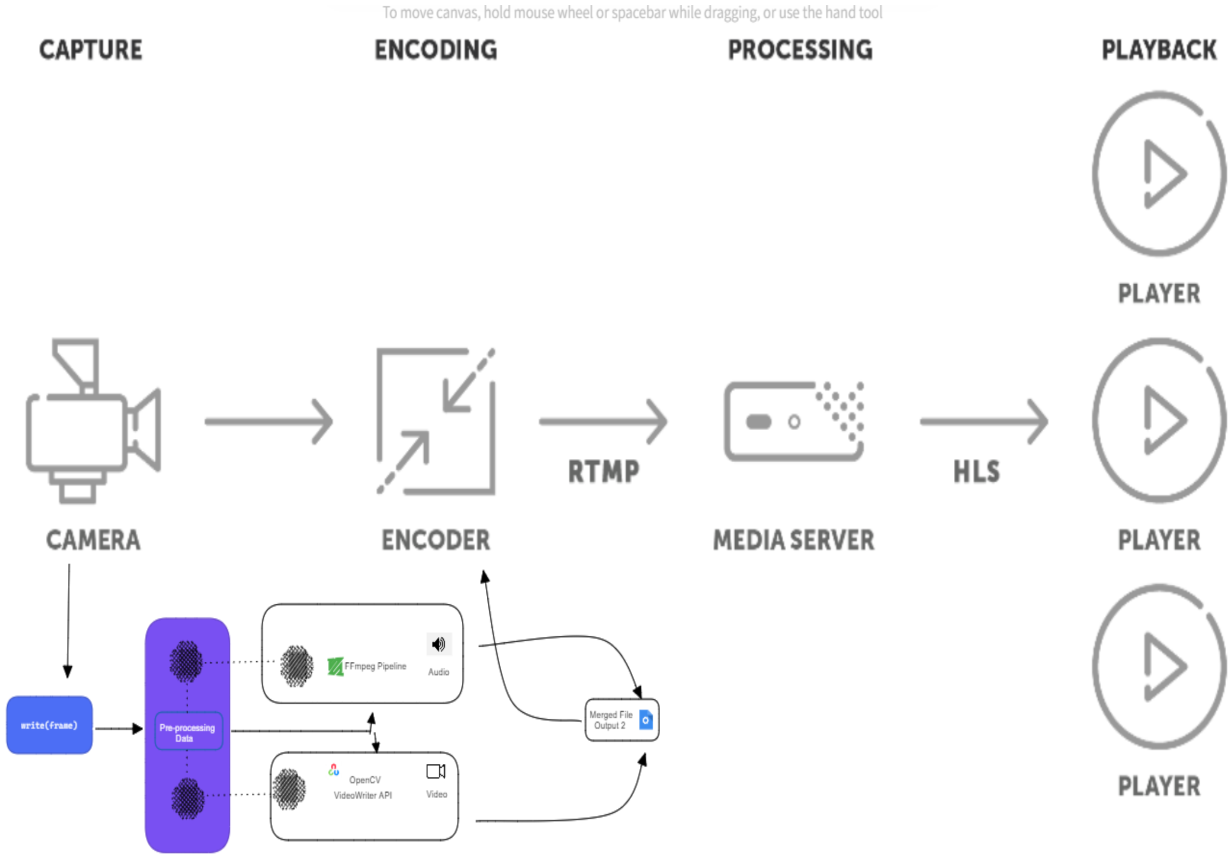
Architecture and Design

- **Flask**: Flask is a web framework used to develop server-side web applications in Python. It have been used to integrate OpenCV and FFmpeg with the streaming server.
- **RTMP and HLS**: These are protocols used for video streaming. RTMP (Real-Time Messaging Protocol) is used for low-latency streaming, while HLS (HTTP Live Streaming) is used for adaptive bitrate streaming.
- **Nginx server**: Nginx is a popular web server used to host websites and applications. It has been used to support for RTMP/HLS streaming.

Flowchart Of OpenCV & FFmpeg



RTMP && HLS WorkFlow



Challenges

1. **Bandwidth management**: Streaming video to multiple clients simultaneously requires a lot of bandwidth, and managing this bandwidth can be a challenge.
2. **Synchronization**: Ensuring that all clients receive the same stream at the same time is critical for a seamless experience. Any delay or lag in the stream can cause synchronization issues.
3. **Encoding and decoding**: Encoding and decoding video and audio in real-time for multiple clients can be resource-intensive, requiring specialized hardware or software to handle the load.
4. **Scalability:** As the number of clients increases, the system must be able to scale to handle the increased load without sacrificing performance.

Best practices:

1. Use adaptive bitrate streaming to ensure that clients receive the highest quality stream possible based on their available bandwidth.
2. Implement caching and load balancing to improve performance and reduce bandwidth usage.
3. Regularly monitor server performance, bandwidth usage, and client connection quality to identify and address potential issues.
4. Implement failover mechanisms to ensure uninterrupted streaming in the event of server or network failures.

Results and Analysis

- The proposed approach is tested using Python programming language and a per- sonal computer with an Ryzen 7 4800H CPU @ 2.9 GHz processor and a 16 GB RAM. The operating system is Windows 11. The average latency time for multi- client streaming was 31 seconds.

- In this section, we will discuss results and performance of our technique and demonstrate the working.

Results and Analysis

Fig 1: Home Page

Results and Analysis

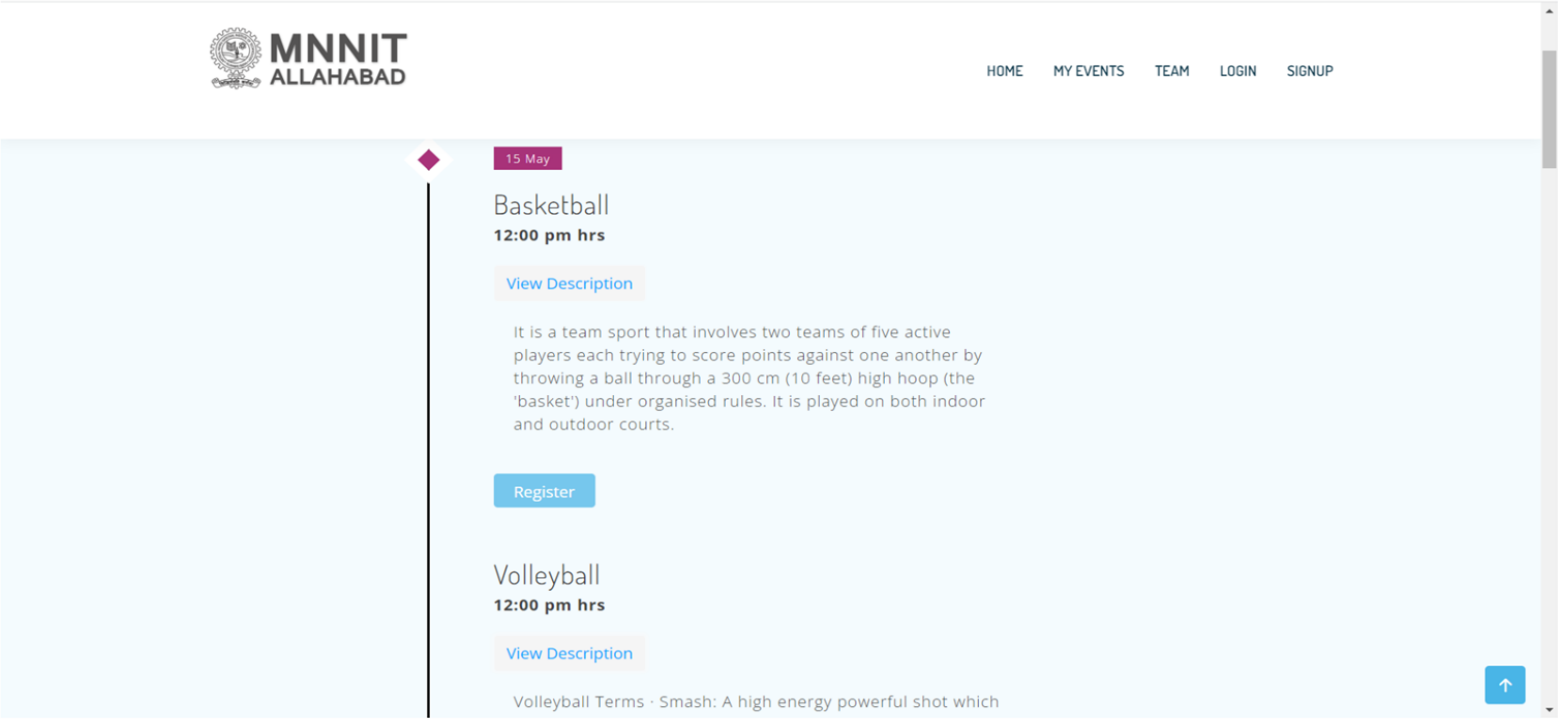


Fig 2: Home Page

Results and Analysis

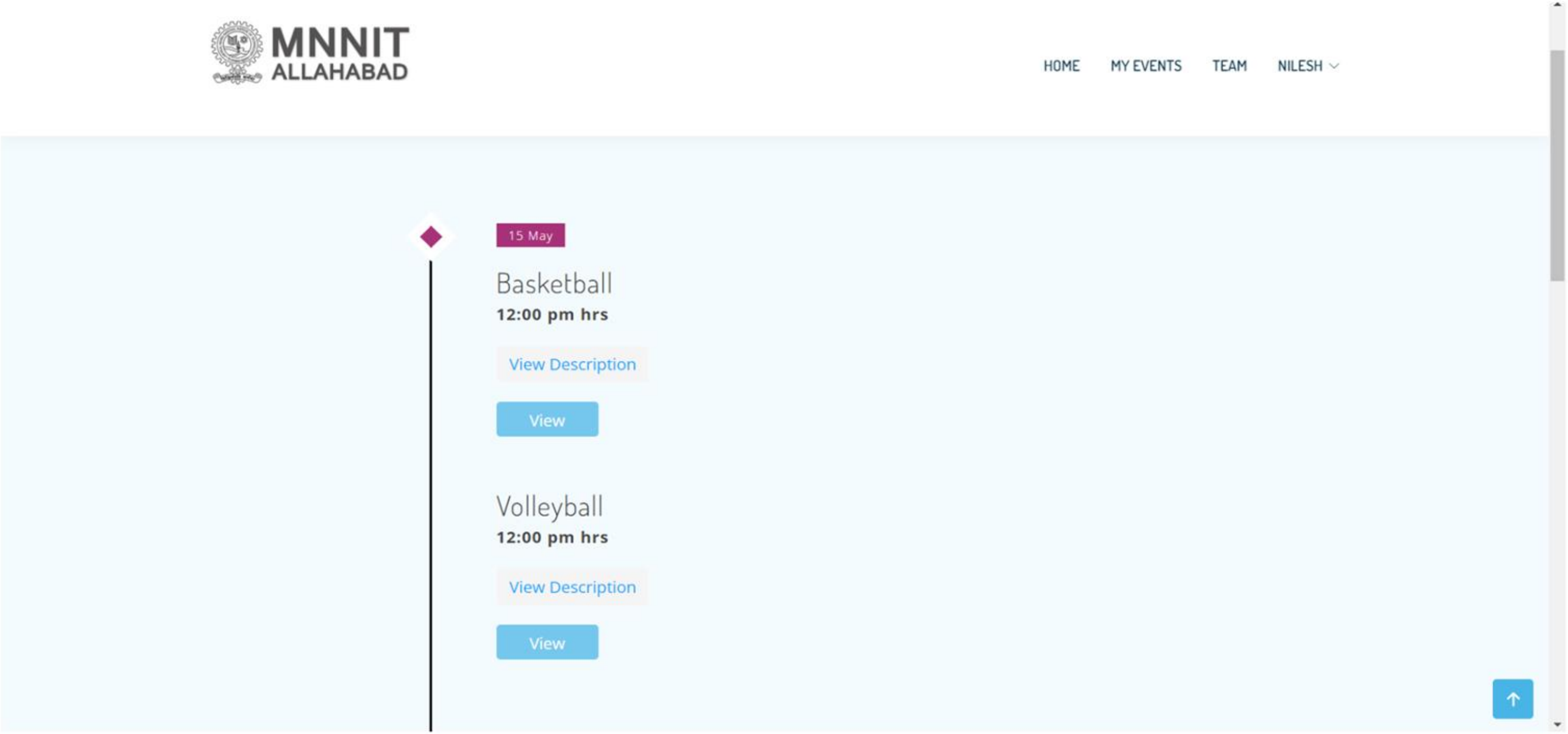


Fig 3: User Events Page

Results and Analysis

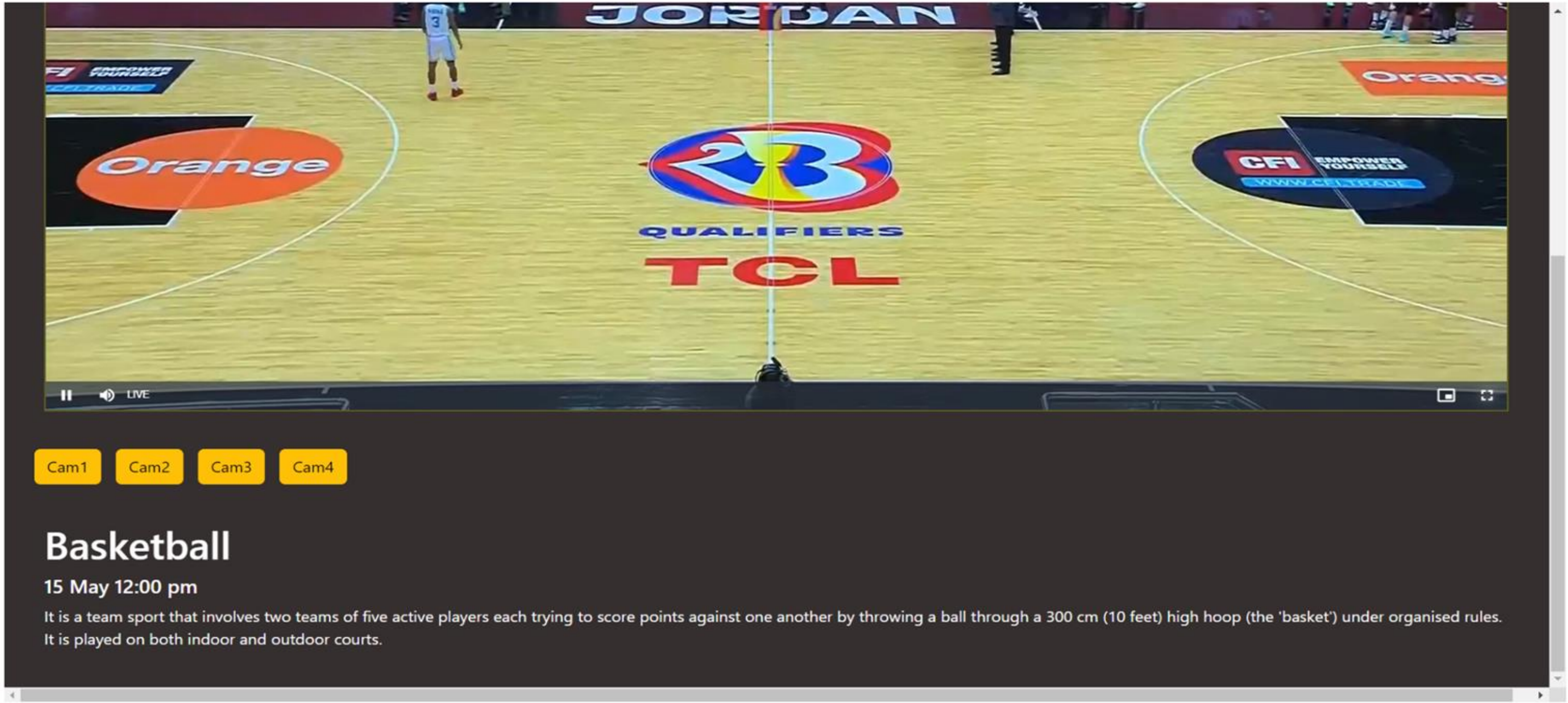


Fig4: Live Stream Page

Results and Analysis

- **Minimum Latency:** Measure the latency of the streaming platform by monitoring the time it takes for a video frame to travel from the source to the client devices. Use tools or methods to measure the round-trip time (RTT) or time taken for a packet to travel from the source to the client and back. Assess the platform's ability to minimize latency by optimizing the streaming pipeline, network configurations, and using low-latency streaming protocols like WebRTC or MPEG-DASH with low-latency modes.

- **Bandwidth:** Conduct bandwidth testing to determine the maximum throughput of the network connection. Use network testing tools or services to measure the upload and down- load speeds available for streaming. Test the streaming platform's ability to adapt to different bandwidth conditions by simulating low-bandwidth or high-bandwidth scenarios and evaluating the platform's performance in delivering video streams with appropriate quality levels.

Results and Analysis

- **Buffer Size:** Test the buffer handling under different net- work conditions, such as varying bandwidth or intermittent connectivity.

Parameter	OpenCV	FFmpeg	OpenMediaStream
Min. Latency	25s	38s	78s
BandWidth	700 - 1000 KBPS	400-500 KKBPS	500 - 800 KBPS
Buffer size	600 - 1000 frame	- - -	- - -

Table1: Parametric Analysis on various technologies

Future Direction

- Reducing latency in multiclient streaming to enhance the real-time experience for viewers by using techniques like Adaptive Bitrate Streaming, Low-Latency Streaming Protocols and Caching and Pre-fetching.
- Enhancing video quality by Video Transcoding and Preprocessing.

Conclusion

- **Efficient Video Streaming**: By leveraging Nginx as a web server and RTMP as a streaming protocol, we have established a robust and scalable infrastructure for delivering video content to multiple clients simultaneously.

- **Flexibility and Compatibility:** With the support of FFmpeg, we have achieved broad compatibility with various video formats and codecs.

- **Adaptive Bitrate Streaming:** The implementation of HLS (HTTP Live Stream- ing) has enabled adaptive bitrate streaming, allowing clients to dynamically adjust the video quality based on their network conditions.

References

1. A. Al-Fuqaha, M. Guizani, M. M. M. A. Multi-client live video streaming: Challenges, approaches, and solutions.’

2. Abdelhak Bentaleb, Zhengdao Zhan, F. T. M. L. S. H. C. T. H. H. R. Z. Low latency live streaming implementation in dash and hls. MultiMedia (2022).

3. S. Liu, L. Jiang, H. W. B. L. S. L. Dynamic adaptive multiclient video streaming for heterogeneous networks.

4. [4] Xiaohua Lei, Xiuhua Jiang, C. W. Design and implementation of streaming media processing software based on rtmp. IEEE (2012).

Thank You !