STREAMING TECHNOLOGIES

COEN 233 - COMPUTER NETWORKS

AUTHOR:
Aishwarya Gupta
07700006172

UNDER THE GUIDANCE OF:
Dr.Keyvan Moataghed

AUDIENCE:	Academic Peers and Instructors, Researchers and Professionals in Computer Science, Technology Enthusiasts and Hardware and Software Industry Practitioners.
WORKING KNOWLEDGE:	Computer Science and Technology Background, Basic Understanding of Computer Networks, Knowledge of Multimedia Basics, Familiarity with Internet Technologies
WHAT IS COVERED IN THIS DOCUMENT:	Introduction to Streaming, Fundamentals of Streaming, On Demand and Live Streaming, Multicast and Unicast, TCP and UDP: A Background, Streaming Protocols including Real-Time Messaging Protocol (RTMP), Real-Time Streaming Protocol (RTSP), HTTP Live Streaming (HLS), Low Latency HLS, Dynamic Adaptive Streaming over HTTP (MPEG-DASH), Low-Latency CMAF with MPEG-DASH, Comparative Study of Streaming Protocols, Video and Audio Compression, Content Delivery Network, Future Trends in Streaming, Conclusion

Table of Contents

Table of Figures

Table of Tables

1. Introduction

In the ever-evolving digital landscape, streaming technologies have emerged as the linchpin of our connected world, fundamentally altering the way we access and experience audio and video content. Streaming technologies consist of the methods and protocols used to transmit multimedia data, such as audio, video, and other forms of data, over computer networks to end users. These technologies enable users to access multimedia content in real-time or on-demand through the Internet or local networks. *Streaming* describes how the content is delivered, rather than about the content itself. As the demand for high-quality, on-demand content continues to surge, the intricate interplay of streaming protocols, audio and video compression techniques, and the inexorable march of technological progress in the streaming industry becomes increasingly central to our digital existence.

This research paper embarks on a journey through streaming technologies, where we will traverse the Fundamentals of Streaming Technologies, deep dive into numerous Streaming Protocols, discuss Content Delivery Networks and understand audio and video compression, and peer into the future trends that will shape the streaming landscape in the coming years. Through this exploration, I aim to demystify the intricacies of streaming, shedding light on the technology that powers our favourite content and the innovations that promise to redefine our viewing and listening experiences.

As streaming rises above mere convenience and matures into a cultural phenomenon, our research takes on renewed significance. The transformative power of streaming is not confined to entertainment alone; it influences education, commerce, communication, and beyond. It's a testament to the adaptive nature of technology and its ability to shape our world. Our research attempts to explore this multifaceted evolution, charting the course of streaming from its past to the horizon of its future.

1.1 Chapter Organisation:

Chapter 1 consists of the Introduction to Streaming Technologies and a brief about what thai research paper will cover.
Chapter 2 consists of fundamentals and terminologies of streaming workflow, like on-demand and live streaming, and unicast and multicast streaming.
Chapter 3 consists of background on TCP and UDP, and elaboration on Streaming Protocols, their Handshake process, their Workflow and Advantages and Disadvantages for RTMP, RTSP and RTP, HLS, Low Latency HLS, MPEG-HASH, and CMAF with MPEG-HASH.
Chapter 4 consists of details about the Video and Audio Compression Technologies such as Lossy and Lossless Compression
Chapter 5 consists of a deep dive into Content Delivery Networks
Chapter 6 consists of a brief discussion on the Future Trends in Streaming Technology

2. Fundamentals of Streaming Technologies

Flow or Streaming is defined very broadly and mainly refers to the network multimedia data transmission technology. This chapter will talk about the generalized multimedia data flow on the Network that is a stable and continuous transmission and real-time playback. Through the streaming media technology, the server can guarantee to the client, the stability and continuity of the multimedia data stream. When this multimedia data is broadcast to the client, the rest of the document will

continue to download from the streaming media server. The client will receive data at a steady rate, instead of waiting for the data to be downloaded and then played back.

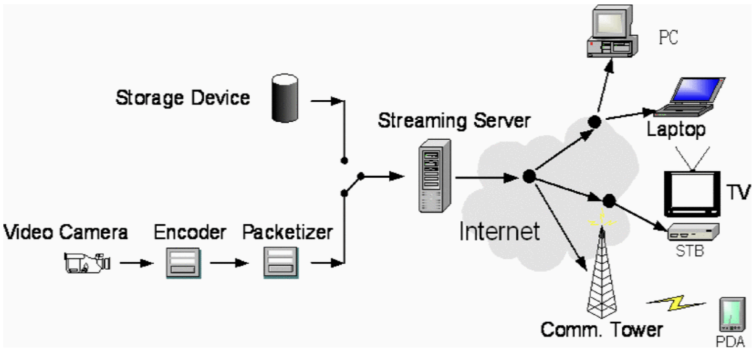

Fig 2.1: Transmission processes of the streaming media

The network architecture for content streaming over the internet involves a combination of components and technologies that deliver content efficiently to end-users. Below is an overview of the key elements of network architecture for content streaming:

Content Source:This is the origin of the multimedia content. It can be a media server, a content distribution server, or a cloud-based content repository. Content is typically stored and managed here.

Content Encoding and Transcoding:Before streaming, multimedia content is often encoded and transcoded into multiple bitrates and formats to support adaptive streaming. This step ensures compatibility with a wide range of devices and network conditions.

Content Delivery Servers:These are responsible for delivering content to end-users. Content delivery servers are typically distributed in various geographical locations to reduce latency and improve reliability. They can be part of a Content Delivery Network (CDN) or a cloud-based infrastructure.

Load Balancers:Load balancers distribute user requests across multiple content delivery servers, ensuring that the streaming system can handle a high volume of users and maintain consistent quality.

Streaming Protocols:Streaming protocols like HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (MPEG-DASH), Real-Time Messaging Protocol (RTMP), and others are used to manage how content is segmented, transmitted, and played back on the user's device.

Quality of Service (QoS) Management:QoS management components ensure a smooth user experience by monitoring and optimizing network conditions in real-time. This may involve adaptive streaming, buffer management, and bitrate adaptation.

Content Security:Encryption and digital rights management (DRM) are employed to protect content from unauthorized access, downloading, or distribution. Security measures also include secure sockets layer (SSL) for data encryption.

User Authentication and Authorization:Access to premium or protected content is typically controlled through user authentication and authorization systems, which verify user identities and permissions.

Client Devices:Users access content through a variety of devices, including computers, smartphones, tablets, smart TVs, and set-top boxes. The device's capabilities, including screen resolution, audio, and video codecs, influence the streaming experience.

Media Players:Media players, such as web browsers, dedicated streaming apps, or hardware media players, are used to decode and display multimedia content on the user's device.

End-User Networks: These networks, including home Wi-Fi, cellular data networks, or broadband connections, determine the quality of the streaming experience. Network congestion, bandwidth, and latency can impact content delivery.

Edge Computing: Edge computing involves deploying computing resources closer to the end-user, reducing latency and improving real-time interactions in streaming applications, especially for live events.

Content Analytics and Monitoring:These components track user interactions, content performance, and system health. Real-time analytics help streaming providers make adjustments and optimize content delivery.

Content Delivery Optimization: This includes mechanisms to optimize content caching, prefetching, and predictive content delivery to reduce latency and improve efficiency.

Network Backbone: High-speed internet infrastructure, often managed by internet service providers (ISPs), forms the backbone that connects the various components of the network architecture.

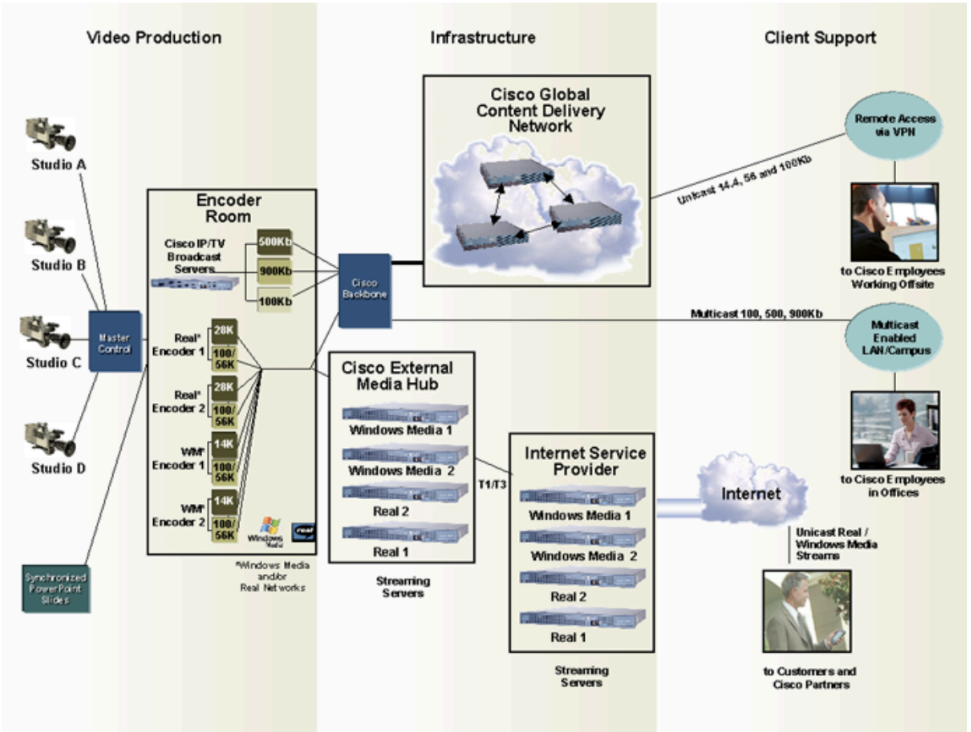

Fig 2.0.2 Overview of Cisco’s Video Streaming Architecture using Unicast and Multicast Streams

2.1 On Demand and Live Streaming

Live streaming is a method of media distribution that provides content to viewers as it is generated. Twitch, Instagram Live and Facebook Live use this streaming method.
On-demand video streaming, also called "video on demand (VOD)," is a media streaming method that allows customers to pick content to watch when required . Youtube is a prime example of video on demand streaming.
Here are the key differences between them -

Aspect	Video on Demand (VOD)	Live Streaming
Content Delivery Timing	Pre-recorded content, delivered at any time.	Real-time content delivery.
Content Storage and Retrieval	Content stored on servers, retrieved on user request.	Content generated and delivered live.
Protocols and Segmentation	Often uses adaptive streaming protocols (e.g., HLS, MPEG-DASH) with segmented content.	May use real-time protocols like RTMP, with continuous delivery.
Latency	Tolerates higher latency; buffering common.	Aims for minimal latency for real-time experience.
Content Libraries and Catalogs	Extensive libraries of content, with user choice.	Focuses on specific live events or channels.
Continuous vs. On-Demand Playback	Users control content playback, with start, pause, rewind, and stop.	Continuous playback with limited user control.
Audience Scalability	High concurrency; scalability is predictable.	Must handle sudden spikes in viewer concurrency during live events.

Table 2.1.1 Difference between VOD and Live Streaming

2.2 Unicast and Multicast

For transmitting data, there are various methods to choose from, consisting of unicast and multicast. Each streaming method has its own advantages and disadvantages, therefore its important to understand the details -

2.2.1 Unicast Streaming

Unicast streaming is a one-to-one connection between a server and a client.Each client is allocated a unique data stream, and only those clients who actively request the stream will be provided with it. Unicast streaming works for both live streaming and on-demand streaming.
How it Works -

- A user initiates a request for information from a server, website, or another user.
- The other party sends the information after establishing a unique connection.
- Unicast utilises User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) .

When a live event is streamed through an OTT app, a server at origin is used to live stream this content to thousands of individual users around the globe. Every user viewing the live stream has a unique connection between their device and the origin server in unicast mode.

Since unicast streaming happens over the internet, network paths and bandwidth availability differ greatly from stream to stream. Since a single origin server would struggle to deliver thousands or even millions of unique unicast streams, Content Distribution Networks (CDN) are used to distribute the streams across all global locations and share the load.
Here are some examples of unicast streaming: Netflix, Hulu, Amazon Prime Video, YouTube, Skype.

2.2.2 Multicast Streaming

Multicast streaming is a one-to-many streaming method that transmits data to multiple devices simultaneously. It's similar to traditional broadcast. Multicast uses UDP (User Datagram Protocol) to "broadcast" a stream over a closed IP network such as a LAN (Local Area Network) or an IP Service provider's own network. In a multicast IP network, the content sender only delivers a single stream, and the nodes throughout the network will replicate this stream across the entire network, similar to a relay race. Using multicast to distribute multimedia to hundreds or thousands of users holds preference for organizations to avoid flooding the network with duplicate streams.

- **Efficient**: Multicast is an efficient way to transmit data to multiple receivers.
- **Bandwidth optimization**: Multicast delivers one single version of a data file to hundreds or thousands of users simultaneously. This reduces network traffic when transmitting large amounts of data.

Here are some examples of multicast streaming:

- IPTV: A centralized server sends a single stream to many endpoints.
- Video server: A video server can send out networked TV channels.
- VLC mediaplayer: This media player can stream and receive videos on the network.
- Online gaming: Multicast transmits data to a specific group of members.

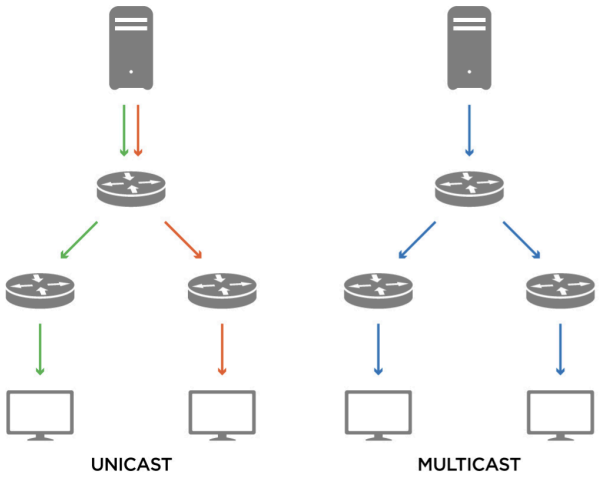

Fig 2.1.1 Unicast vs Multicast Streaming

3 Streaming Protocols

Every time we view a live stream or video on demand(VOD), streaming protocols are utilised to deliver data over the internet. These can be present in the application, presentation, or session layers of the ISO OSI Model.

3.1 TCP and UDP: A Quick Background

TCP (Transmission Control Protocol) is a transport layer protocol and its primary role involves ensuring the connection oriented and reliable transmission of data between network-connected devices, primarily over the internet. TCP consists of error-checking mechanisms, making it suitable for applications where data integrity is critical. It was designed to deliver secure end-to-end data stream across potentially unreliable network environments.
TCP requires an initial connection setup between hosts, and it has the ability to identify problems with IP packets, whether arising from network congestion, load balancing challenges, or unusual network anomalies. When TCP detects missing, duplicated, or out-of-sequence IP packets, it initiates a request for retransmission of the missing data, rectifies the order of out-of-sequence data, and takes measures to reduce network congestion. Once the packets are correctly arranged in the intended order, TCP forwards the data to the recipient application.[13]

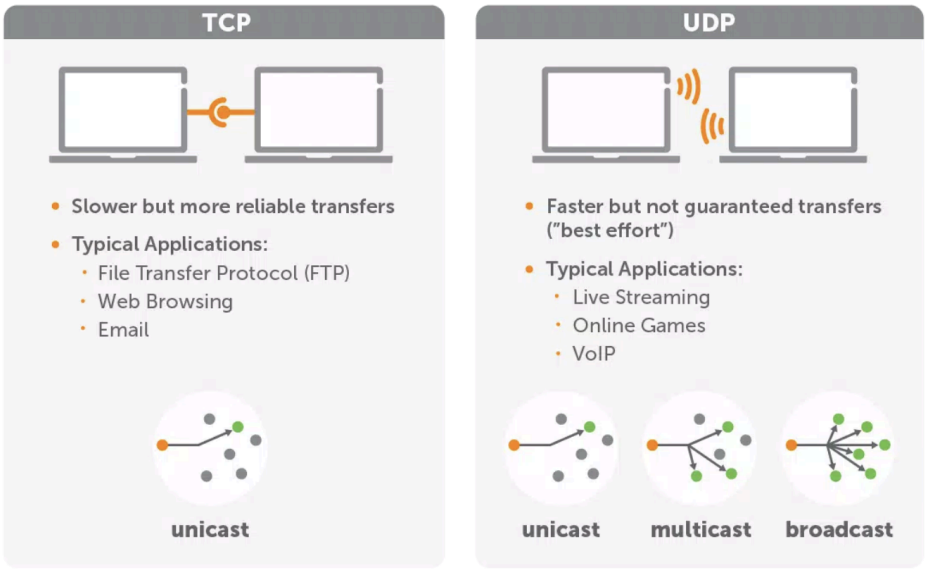

Fig 3.1.1 Applications for TCP and UDP

UDP (User Datagram Protocol) is a lightweight and connectionless network protocol that prioritizes speed and low latency over reliability. It is simple and efficient, allowing for fast data transmission but not guaranteeing data integrity.UDP relies on checksums instead of handshakes to validate data and provides no acknowledgement of delivery, and ordering. It's also stateless, which makes it ideal for transmitting data to large numbers of clients for real-time applications like online gaming, video conferencing, and live streaming, where small delays are acceptable.As such, UDP is ideal to use alongside Real-Time Streaming Protocol (RTSP).

As HTTP communications generally take place on TCP connections, we see HTTP-based streaming protocols over TCP. However, latest technologies leverage UDP's speed and simplicity. The below chart demonstrates which streaming protocols pair with TCP or UDP:

TCP	UDP
RTMP (Real-Time Messaging Protocol)	SRT (Secure Reliable Transport)
RTSP (Real-Time Streaming Protocol)	WebRTC (Web Real-Time Communications)
Apple HLS (includes Low-Latency HLS)	RTSP (Real-Time Streaming Protocol)
MPEG-DASH (includes CMAF)	RTP (Real-Time Transport Protocol)

Table 3.1.1 Streaming protocols paired with TCP and UDP

3.2 Real-Time Messaging Protocol (RTMP)

Real-Time Messaging Protocol (RTMP) is an application layer protocol which is utilised to stream data, audio, and video over the Internet. It operates over TCP and was developed originally as a proprietary protocol by Macromedia to stream between Flash Player(Client) and the Flash Communication Server, Adobe then acquired Macromedia and used it in applications including Adobe LiveCycle Data Services ES. They released a version of the protocol for public utilisation.[01]

RTMP is used for real-time, low-latency multimedia streaming and interactive applications where maintaining a timely and continuous flow of data is crucial.

Audio Codecs	AAC, AAC-LC, HE-AAC+ v1 & v2, MP3, Speex
Video Codecs	H.264, VP8, VP6, Sorenson Spark®, Screen Video v1 & v2
Playback Compatibility	Not widely supported Only limited use in Flash Player (discontinued now by Adobe), Adobe AIR, RTMP-compatible players It is no longer accepted by most browsers, Android, iOS and most players
Benefits	Low latency, adaptable(rewind,pause etc) and minimal buffering
Drawbacks	It is no longer updated or supported
Variant Format	RTMP, works along with Transmission Control Protocol (TCP) RTMPS, RTMP uses TLS/SSL(Transport Layer Security) connection. RTMPT, encapsulated in HTTP requests to go through firewalls and traffic filtering mechanisms. This session can carry plain RTMP, RTMPS, or RTMPE packets inside. RTMFP, using UDP and allows users to connect directly with one another (Point to Point).
Latency	5 seconds

Table 3.2.1 RTMP Snapshot

3.2.1 Handshake and Connection Setup

RTMP utilises a handshake process for client server connection. The handshake consists of three phases: version negotiation, challenge, and response.

1. Version Negotiation : The RTMP handshake begins with version negotiation, where the client and server exchange information about the RTMP version they intend to use.The client sends a C0 message, which contains the version of RTMP it supports. The server responds with an S0 message, indicating its version preference. If there is a version mismatch, they may need to negotiate a compatible version or terminate the connection.
2. Challenge and Response : In the challenge and response phase, the server sends a random data challenge to the client in an S1 message. The client then responds with a calculated S2 message. The S1 message from the server is cryptographically random data, and the S2 message from the client is derived from the challenge data. This step allows both parties to have access to the cryptographic keys to carry out secure communication.

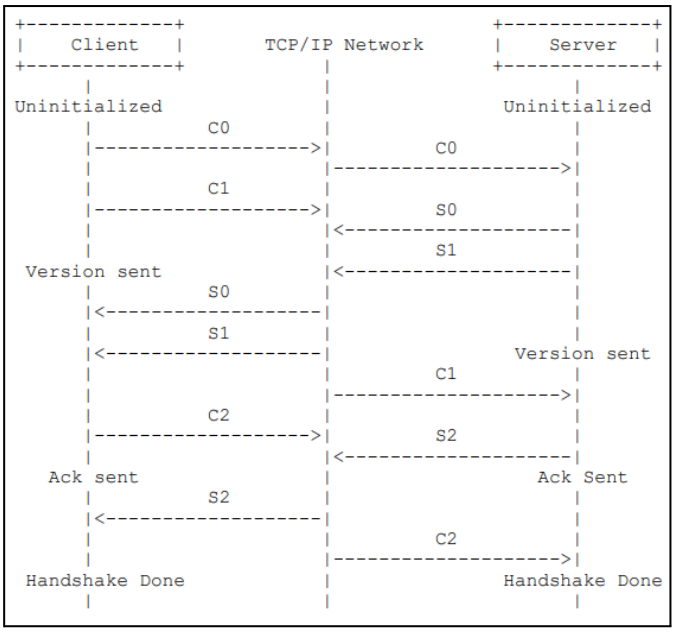

Fig 3.2.1 RTMP Handshake

3. Acknowledgement and Confirmation: Once the S2 message is received by the server, it verifies the client's response. If the verification is successful, the server sends an acknowledgement in an S3 message to confirm the completion of the handshake.The client receives the S3 message and acknowledges it with a C2 message.This mutual acknowledgement marks the successful completion of the handshake, and the connection is considered established.

3.2.2 Connection Channels

RTMP establishes several channels or streams used for communication within the RTMP protocol. These channels enable the simultaneous transmission of data including control messages, audio data, and video data.
1. **Control Channel:** The control channel is responsible for managing and controlling the protocol's communication. It carries control messages including information about the session, handshaking, protocol version negotiation, and other administrative data.
2. **Audio Channel:** In the context of multimedia streaming, this channel allows for the real-time transmission of audio content, which is crucial for applications such as online radio, video conferencing, and live broadcasting.The audio channel supports various audio codecs, allowing flexibility in encoding and decoding audio data.
3. **Video Channel:** This channel enables the real-time delivery of video content, making it suitable for applications like live video streaming, video conferencing, and interactive gaming. Like the audio

channel, the video channel supports different video codecs to accommodate various video formats and qualities.
In an RTMP session, multiple channels are active at the same time. When encoding data in RTMP, it involves the creation of a packet header.

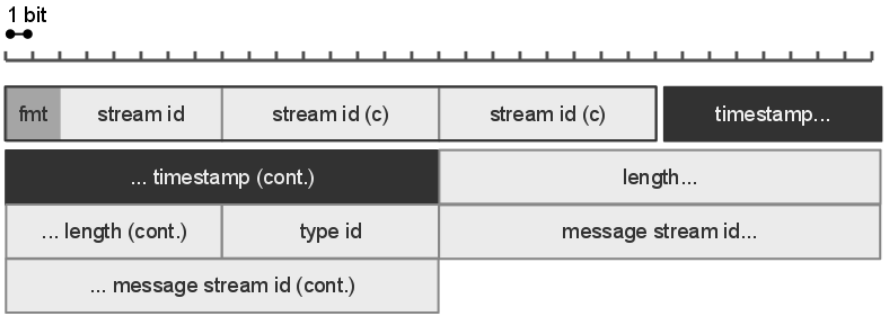

Fig 3.2.2 RTMP Packet Diagram

3.2.3 RTMP Packet Configuration

1. **Header**: Key header fields include the channel ID, timestamp, packet length, message type, and message stream ID.The channel ID identifies the specific channel on which the packet is transmitted.The timestamp indicates when the packet was generated, particularly relevant for synchronization in multimedia streaming.The message type distinguishes the purpose of the packet, such as audio, video, control, or metadata.The message stream ID helps identify the stream to which the packet belongs.
2. **Payload**: The packet length specifies the size of the packet's payload.For audio and video packets, the payload contains the encoded audio or video data.Control packets carry various control information, such as acknowledgement messages or session setup details.Metadata packets may contain information about the stream, including stream name or metadata about the media.
3. **Chunking**: RTMP allows for the fragmentation of large payloads into smaller chunks, each transmitted as a separate packet.The chunking process involves dividing the payload into smaller fragments, with each fragment having its header indicating its position within the larger payload.

3.2.4 RTMP Streaming Workflow

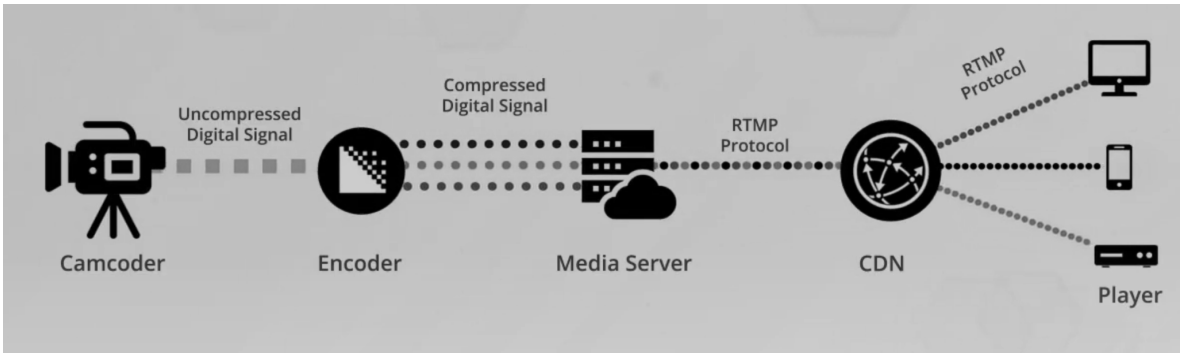

Fig 3.2.3 RTMP Streaming Diagram

1. The real-time multimedia content which is captured by cameras, or professional equipment are encoded into digital signals , and are compressed for transmission.
2. The broadcaster sets up an RTMP server by defining essential configurations. The Broadcasting software, like Wirecast, is set up to configure a connection to the RTMP server. This software is responsible for the encoding, packet formation, and transmission of the real-time content.

3. The broadcaster initialises a live stream, over the connection setup with the RTMP server. The server will then receive this encoded data and it configures it for distribution.
4. During the live multimedia streaming, viewers have the opportunity to actively participate in the content and engage with other participants using features like chat, comments, likes, and interactive overlays. These interactive elements enhance the viewing experience, making it dynamic and immersive.[01]

Modern HTML5 players rely on HTTP-based protocols like HLS, which means that a media server or streaming service can be employed to receive an RTMP stream and convert it into a more suitable format for playback.

3.2.5 Advantages and Disadvantages

Advantages

Low Latency: RTMP guarantees a stable connection for live video streams, ensuring a consistent viewing experience for users, even in the presence of an unreliable internet connection.
Adaptability and Flexibility: With the RTMP protocol, developers have the capability to seamlessly integrate a range of video formats, including audio, video, and text, into a unified and cohesive package.

Disadvantages

Limited Support: The Flash format is rapidly becoming outdated, with HTML5 players replacing it. RTMP, which is currently supported by Flash, cannot be played on HTML5 players without a converter, such as an HTTP-based video protocol.
Low Bandwidth: RTMP streams are susceptible to bandwidth problems, resulting in frequent and frustrating interruptions in live streams that can have a detrimental effect on the user experience (UX).

3.3 Real-Time Streaming Protocol (RTSP)

The Real Time Streaming Protocol (RTSP) is an application layer protocol that offers a comprehensive framework for controlling and delivering real-time data on demand. This includes various types of data like audio, video, and text. RTSP is versatile and supports the transmission of both live, real-time data and stored recordings. It's capable of managing multiple sessions for data delivery and allows users to select delivery channels such as UDP multicast, UDP, or TCP, depending on their specific needs. Additionally, RTSP provides a mechanism for choosing delivery methods based on the Real-time Transport Protocol (RTP), which is often used for streaming multimedia content.

When users begin a video stream using a camera that utilises RTSP, the device will send an RTSP request to the streaming server, and initialise the setup. Following this setup, the video and audio data can be transmitted using RTP. In this analogy, you can envision RTSP as the equivalent of a television remote control, facilitating the control and coordination of media streaming, while RTP serves as the actual broadcast medium, responsible for transmitting the audio and video content.[13]

Audio Codecs	AAC, AAC-LC, HE-AAC+ v1 & v2, MP3, Speex, Opus, Vorbis
Video Codecs	H.265 (preview), H.264, VP9, VP8
Playback Compatibility	Seldom used for playback (Quicktime Player and other RTSP/RTP-compliant players, VideoLAN VLC media player, 3Gpp-compatible mobile devices)
Benefits	Low-latency and ubiquitous in IP cameras
Drawbacks	Not optimized for quality of experience and scalability
Variant Format	RTSP as an umbrella term describes the entire stack of RTP, RTCP (Real-Time Control Protocol), RTSPS (RTSP over SSL / Secure RTSP), and good-old RTSP
Latency	2 seconds

Table 3.3.1 RTSP Snapshot

3.3.1 RTSP Handshake

RTSP (Real Time Streaming Protocol) establishes a client-server connection through a series of request and response messages between the client (often a media player or device) and the server (the streaming source). Here's a simplified overview of how the connection is established:

Client Request: The process begins when the client sends an RTSP request to the server. This request specifies the type of operation to be performed, such as "PLAY," "SETUP," or "DESCRIBE," and includes information about the media stream the client wants to access. For example, the client may request a specific video stream or audio stream.

Server Response: Upon receiving the client's request, the server processes the request and responds with an RTSP response. This response provides information about the status of the request and any additional details needed for the client to proceed.

Setup Phase: During the setup phase, the client and server negotiate the parameters for the media stream. This includes selecting the transport protocol (e.g., UDP or TCP), port numbers, and other settings necessary for data transmission. The server informs the client about the chosen transport protocol and setup details.

3.3.2 RSTP Requests

- Options: Request that determines the types of requests that the server accepts.
- Describe: This request determines the URL and data type.
- Announce: This method defines the presentation when it is being sent to the server from the client and it revises the description when sent to the client from the server.
- Setup: This request specifies the transportation of a media stream before sending a play request.
- Play: This request starts the transmission of media by informing the server to begin sending the multimedia data.
- Pause: This request halts the stream delivery temporarily.
- Record: This request triggers the recording of media.

- Redirect: These requests notify the client to initialise the connection to another server by supplying a new URL to issue requests.

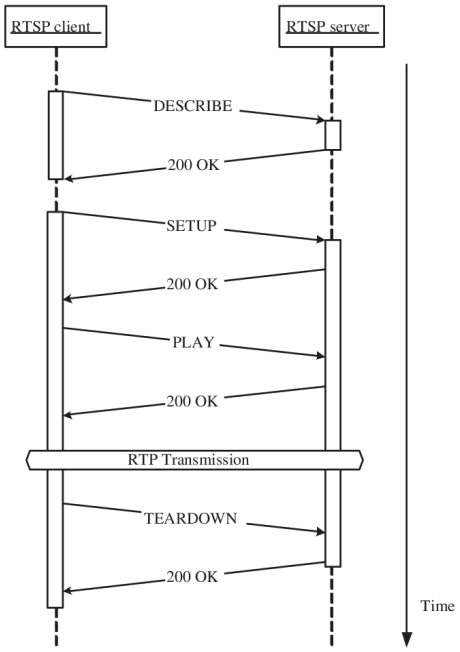

Fig 3.3.1 RTSP Handshake

3.3.3 RSTP Streaming Workflow

Once the setup is complete, the client can send a "PLAY" or "RECORD" request to instruct the server to start streaming the media. If the client wants to record the stream, it will send a "RECORD" request. The server acknowledges these requests.

Data Transmission: With the connection established and the media stream parameters configured, the server begins transmitting the actual audio and video data using the chosen transport protocol (usually RTP - Real-time Transport Protocol). The client receives and processes the media data for playback or recording.

Teardown and Termination: When the client no longer requires the media stream, it can send a "TEARDOWN" request to the server. This request signals the termination of the connection and the release of resources.[2]

It's important to note that RTSP is a control protocol and does not carry the actual media data. The media data is typically transmitted using RTP, and RTSP is responsible for negotiating the setup, control, and termination of the streaming session. This separation of control and data transmission allows for flexibility in choosing different transport protocols and codecs for the media streams.

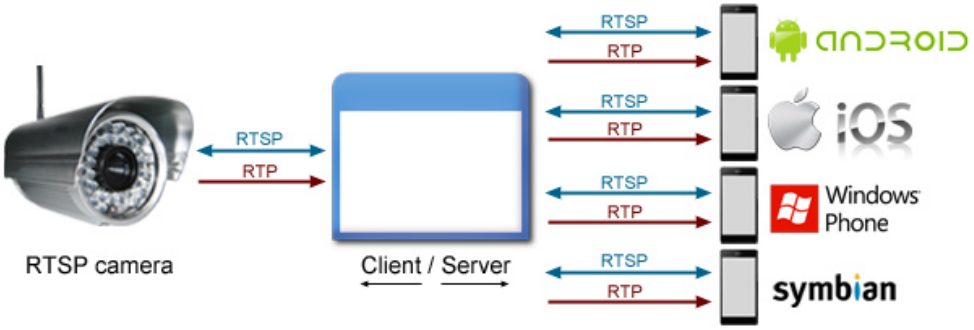

Fig 3.3.2 Streaming using RTP and RTSP

3.3.4 RTP with RTSP

RTP (Real-time Transport Protocol) is a network protocol used in conjunction with RTSP (Real-Time Streaming Protocol) to transmit real-time multimedia data, such as audio and video streams. RTP is designed to ensure the efficient and timely delivery of multimedia content, making it suitable for applications like video conferencing, live streaming, and online gaming.

Here are some key features and functions of RTP:
Media Transport: RTP is responsible for transporting audio and video data over the network. It does not focus on session control or setup, which is the role of RTSP.

Timestamping: RTP assigns a timestamp to each multimedia packet to maintain synchronization between different types of media, such as audio and video. This guarantees the synchronisation of the audio and the video during playback.
Sequence Numbers: RTP assigns a sequence number to each packet, allowing the receiver to reorder packets and recover lost ones.
Payload Types: RTP specifies the type of media payload carried in each packet, helping the receiver understand how to decode and render the data.
RTCP (RTP Control Protocol): In addition to RTP, RTCP is used to provide control and feedback information about the quality and health of the media stream. It is used for functions like reporting on packet loss and jitter.
Real-Time Delivery: RTP is designed for low-latency, real-time delivery of multimedia content. It is sensitive to timing constraints and aims to minimize delays.

3.3.4.1 RTP Packet

RTP Header:
Version (V): A 2-bit field indicating the version of the RTP protocol in use.
Padding (P): A 1-bit flag that indicates whether there is padding at the end of the packet. Padding is used to align the packet to a specific boundary.
Extension (X): A 1-bit flag indicating the presence of an extension header.
CSRC Count: A 4-bit field specifying the number of contributing sources in the packet.
Marker (M): A 1-bit marker that can be used for various purposes, such as indicating the start of a new frame or event.
Payload Type: An 8-bit field indicating the type of media payload (audio, video, etc.) carried in the packet.
Sequence Number: A 16-bit field that helps the receiver reorder packets and detect packet loss.
Timestamp: A 32-bit timestamp value, which allows the receiver to synchronize audio and video streams.
SSRC (Synchronization Source Identifier): A 32-bit identifier that uniquely identifies the source of the RTP stream.
Contributing Source (CSRC) List: This is a variable-length list of 32-bit identifiers for the contributing sources. The number of CSRC identifiers is determined by the CSRC Count field in the RTP header.

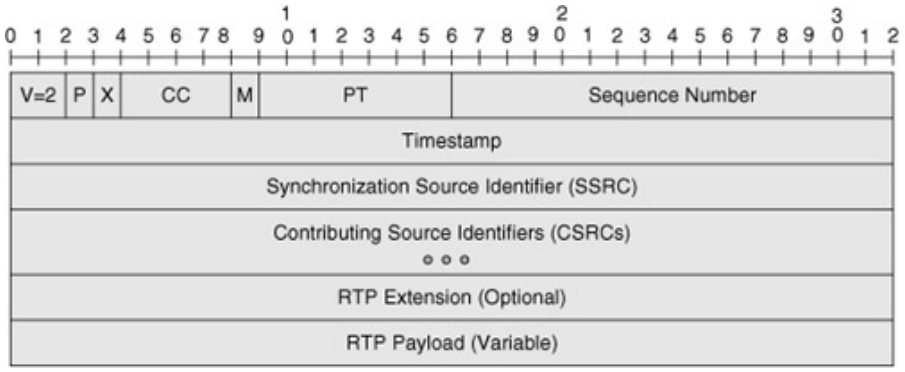

Fig 3.3.3 RTP Packet Header details

3.3.5 Advantages and Disadvantages

Advantages-

Streaming in Segments: Viewers can start watching content from the RTSP stream without having to download the entire video beforehand.

Highly Customizable: You have the option to develop your own video streaming applications by making use of alternative protocols like Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Disadvantages-

Low Popularity: RTSP is notably less commonly used compared to the other protocols listed, primarily because most video players and streaming services do not offer support for RTSP streaming.

HTTP Incompatible: Streaming RTSP directly over HTTP is not a straightforward process, which implies that there is no convenient method for streaming RTSP in a web browser.

3.4 HTTP Live Streaming (HLS)

HLS, or HTTP Live Streaming, is a streaming protocol developed by Apple for the delivery of multimedia content, particularly audio and video, over the Internet. It is designed to work with standard web servers and uses HTTP as the transport protocol. In HLS, video and audio segments, along with the playlist files, are transported over HTTP using TCP connections. HLS is widely used for streaming video content to various devices, including smartphones, tablets, and desktops due to its adaptability to varying network conditions, support for adaptive streaming, and compatibility with a diverse array of devices . It is particularly popular in the context of online video streaming, live broadcasts, and video-on-demand services.

HTTP is universally supported by all internet-connected devices, simplifying its implementation compared to streaming protocols that necessitate specialized servers. An additional advantage is the adaptability of HLS streams, which can dynamically adjust video quality in response to varying network conditions, all without disrupting playback. As a result, video quality might improve or degrade while a user is in the midst of watching. This capability is commonly referred to as **"adaptive bitrate video delivery"** or **"adaptive bitrate streaming."** It is a critical feature, as it prevents slow network conditions from causing video playback interruptions.[3]

Audio Codecs:	AAC-LC, HE-AAC+ v1 & v2, xHE-AAC, Apple Lossless, FLAC Video Codecs: H.265, H.264
Playback Compatibility:	Great (All Google Chrome browsers; Android, Linux, Microsoft, and MacOS devices; several set-top boxes, smart TVs, and other players)
Benefits:	Adaptive bitrate, reliable, and widely supported.
Drawbacks:	Quality of experience is prioritized over low latency.
Latency:	While HLS traditionally delivered latencies of 6-30 seconds, the Low-Latency HLS extension has now been incorporated as a feature set of HLS, promising to deliver sub-2-second latency.

Table 3.4.1 HLS Snapshot

3.4.1 HLS Architecture

In the context of HLS (HTTP Live Streaming), the server, distributor, and client play distinct roles in the delivery of multimedia content to viewers. Here's an explanation of each component:

1. **Server**: The server is the origin point of the multimedia content. It is responsible for hosting the original video and audio files, as well as the segmented media files. The server typically runs on a web server or a content delivery network (CDN). It creates and hosts the manifest files (usually in M3U8 format) that contain information about available streams, URLs to media segments, and metadata.
2. **Distributor** (**CDN**): In some cases, especially for larger-scale deployments, a content distributor is involved in the architecture. This is typically a CDN (Content Delivery Network). The distributor optimizes the delivery of media content to clients by strategically caching and load-balancing content at edge servers located in various regions.
3. **Client**: The client is the end-user's device, which can be a media player, web browser, or a dedicated streaming application. The client initiates the streaming session and plays the multimedia content. Its roles include sending an HTTP request to the server for the manifest file when a viewer decides to watch a video or live stream, analyzing the manifest file to determine the most suitable quality level for the viewer based on their device capabilities and network conditions, and requesting and caching media segments as playback progresses, ensuring smooth, uninterrupted streaming.

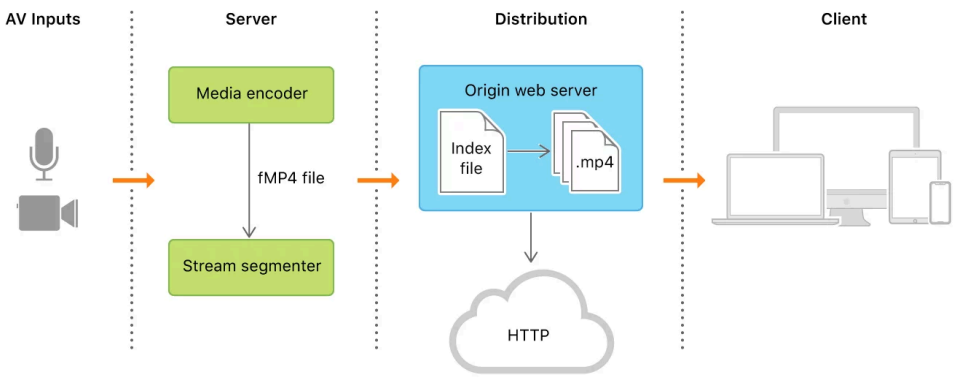

Fig 3.4.1 HLS Streaming Architecture

3.4.2 HLS Handshake and Streaming Workflow

Streaming using HLS involves several key components and steps.

1. **Content Source**: The content source refers to the location where the multimedia content is originated or stored. This can be a live video feed, pre-recorded media, or other sources of audio and video content.
2. **Encoder and Transcoder**: Content is often encoded into various quality levels and formats to support adaptive streaming. Encoders and transcoders prepare the media content for delivery. Common video codecs include H.264 and H.265, while audio codecs like AAC are used.
3. **Segmentation**: The encoded content is divided into smaller segments, typically 2 to 10 seconds each. These segments are stored as media files, often in the MPEG-2 Transport Stream (TS) format.
4. **Manifest File Creation**: A manifest file, typically in the M3U8 format, is generated. This file serves as the playlist that contains information about available streams, their bitrates, and the URLs to the segments.
5. **Content Delivery Server**: The content is hosted on a web server or a content delivery network (CDN). This server handles requests from clients and serves the media segments and manifest files using HTTP.
6. **Client Request**: The client (e.g., a media player or browser) initiates the streaming session by requesting the manifest file over HTTP. The client may include information about its capabilities and network conditions.
7. **Adaptive Streaming**: The client analyzes the manifest file to determine the most appropriate stream quality based on its capabilities and network conditions. It then starts requesting media segments based on its selected stream.
8. **Media Segment Retrieval**:The client fetches media segments (video and audio) and caches them locally for playback. It continually requests the next segments as the content plays.
9. **Playback**:The client plays the content in real-time, ensuring a smooth, uninterrupted viewing experience. It can adaptively switch to different quality levels as network conditions change.
10. **Live Streaming Support**:For live streaming, new segments are continuously generated and added to the server. The manifest file is updated to point to the latest segments in real-time.
11. **Adaptive Bitrate Switching**: The client can dynamically switch between different quality levels to maintain smooth playback. This process is based on real-time evaluation of the viewer's network conditions.
12. **User Interaction**:Viewers can interact with the content, adjusting settings like play, pause, seek, or volume control. These commands are sent to the server using HTTP requests.[14]

3.4.3 Advantages and Disadvantages

One of the major advantages of HLS is its wide compatibility with various devices and firewalls, setting it apart from other streaming formats. However, HLS live streams typically exhibit a latency of 15-30 seconds, which may not meet the requirements of all users. To address this concern, Dacast now provides a direct **low latency streaming feature** for HLS. This feature is compatible with any HLS-compatible encoder and helps reduce the prolonged latency often associated with HLS streaming, offering a quicker streaming solution.

3.5 Low Latency HLS

Low latency HLS (HTTP Live Streaming) is a variation of the HLS protocol that aims to reduce the latency or delay in delivering multimedia content to viewers. Low latency HLS addresses the latency in HLS by optimizing the streaming process to minimize the delay between the moment content is captured or generated and when it is actually viewed by the audience. Here are key characteristics of low latency HLS:

Chunked Segments: Low latency HLS divides media content into smaller, more frequent segments, often called "chunks." These chunks are typically shorter in duration (e.g., 2-4 seconds) compared to the 10-second or longer segments in standard HLS. Smaller chunks allow for quicker delivery to viewers.

Real-Time Processing: The encoding and packaging of media segments occur as close to real-time as possible. This means that content is processed and made available for delivery with minimal delay.

Reduced Buffering: Low latency HLS minimizes the need for buffering on the viewer's end by ensuring that segments are available for immediate playback.

Near-Real-Time Interaction: It enables applications where near-real-time interactions with viewers are crucial. For example, during a live sports event, low latency HLS can provide viewers with a more immediate viewing experience, allowing them to see the action almost as it happens.

Compatibility: Low latency HLS maintains compatibility with HLS-compatible devices and players. This means that it works seamlessly on a wide range of devices, making it accessible to a broad audience.

Adaptive Streaming: Like standard HLS, low latency HLS supports adaptive streaming, allowing the client to switch between different quality levels based on network conditions.[04]

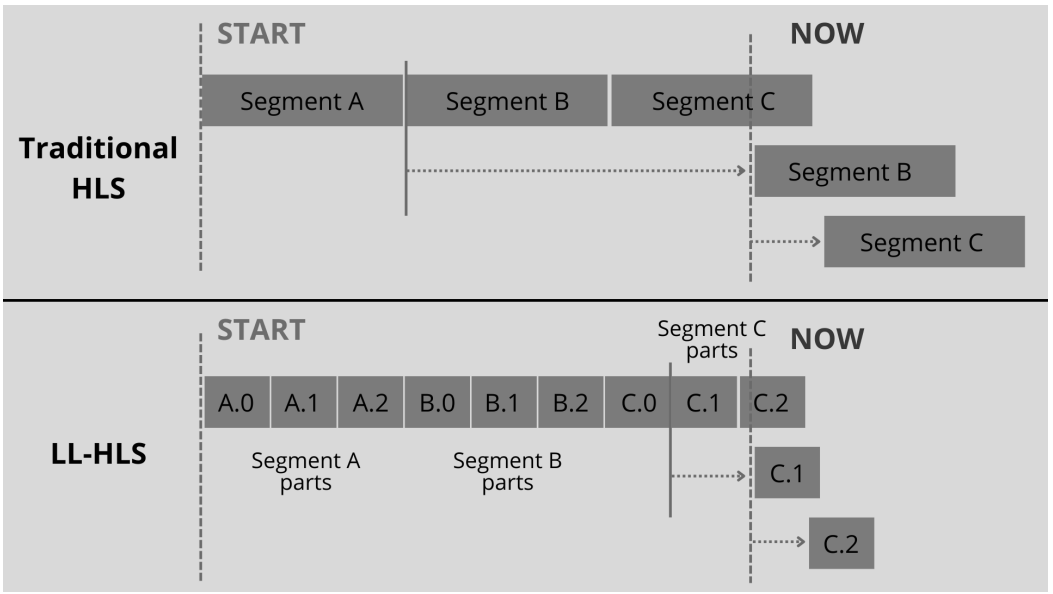

Fig 3.4.2 HLS vs Low Latency HLS

3.6 Dynamic Adaptive Streaming over HTTP (MPEG-DASH)

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) is a streaming protocol and international standard for delivering multimedia content, such as video and audio, over the internet. It is designed to enable adaptive streaming, allowing media content to be delivered efficiently to a wide range of devices with varying network conditions. MPEG-DASH is a part of the MPEG (Moving Picture Experts Group) family of standards and is known for its flexibility and broad compatibility. It uses a Media Presentation Description (MPD) file, which is an XML document that provides information about available streams, their characteristics, and URLs to media segments. The MPD acts as a playlist and guides the client in selecting the appropriate segments for playback. Similar to HLS, MPEG-DASH segments multimedia content into smaller files or chunks. These segments are encoded at various bitrates and resolutions, allowing for smooth quality switching during playback. The client can switch between different quality levels seamlessly as network conditions change, providing a smooth streaming experience. MPEG-DASH is adaptable to low-latency streaming scenarios, is codec-agnostic and can be optimized to reduce playback delay for real-time or interactive applications.[05]

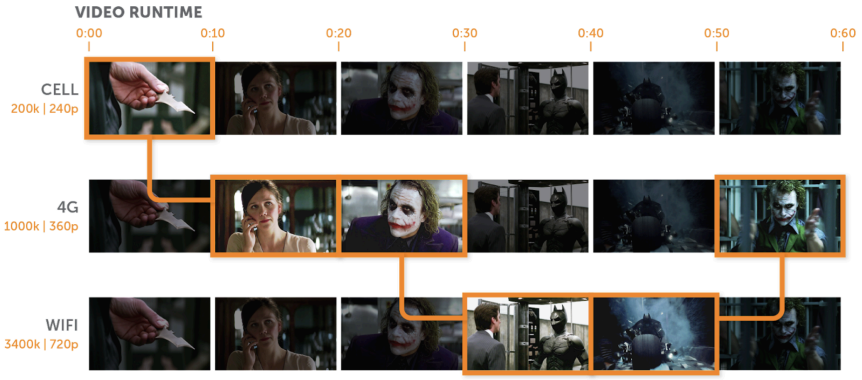

Fig 3.5.1 Adaptive Bitrate Streaming

MPEG-DASH is an international standard (ISO/IEC 23009-1) and is maintained by the MPEG consortium. This standardization ensures consistency and interoperability among different implementations.

Audio and Video Codecs:	Codec-agnostic
Playback Compatibility:	Good, HTML5 video players enable playback on all Android devices; most post-2012 Samsung, Philips, Panasonic, and Sony TVs; Chrome, Safari, and Firefox browsers. iOS and Apple TV do not support DASH
Benefits:	Vendor-independent, international standard for adaptive bitrate
Latency	6-30 seconds (lower latency only possible when tuned or delivered via chunked transfer encoding)
Variant Formats:	MPEG-DASH CENC (Common Encryption)

Table 3.5.1 MPEG-DASH Snapshot

3.6.1 MPEG-DASH Handshake

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) does not involve a formal handshake process like some other protocols, such as TLS/SSL handshakes in HTTPS. Instead, it uses a simple request-response model based on HTTP for communication between the client and the server. Here's how the interaction works without a handshake:

Client Request: When a viewer initiates playback of an MPEG-DASH stream, the client (media player or application) sends an HTTP GET request to the server for the Media Presentation Description (MPD) file, which serves as the playlist for the content.
Server Response: The server responds to the client's request by providing the MPD file. This MPD file contains information about the available streams, their characteristics, and URLs to the media segments. The MPD file is essentially a guide for the client to select the appropriate segments for playback. There is no formal handshake involved in this response.

3.6.2 MPEG-DASH Streaming Workflow

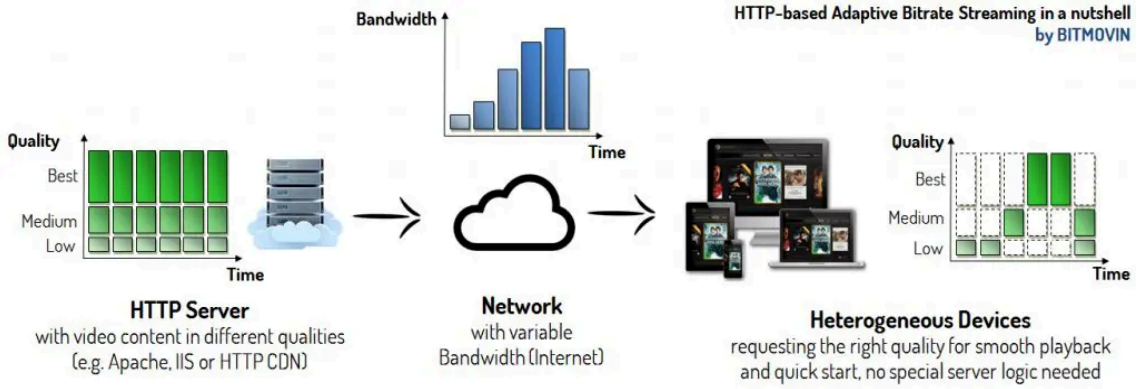

Fig 3.5.2 MPEG-DASH Streaming Workflow

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) works by segmenting multimedia content into smaller pieces and delivering them over HTTP to enable adaptive streaming. Here's how MPEG-DASH works:

1. **Content Encoding and Segmentation**: The multimedia content, such as video and audio, is first encoded into various quality levels or bitrates using codecs like H.264, H.265 (HEVC), VP9 for video, and AAC for audio. The content is then segmented into smaller files, typically 2 to 10 seconds in duration. These segments are encoded at different bitrates and resolutions to cater to a range of network conditions and device capabilities.
2. **Media Presentation Description (MPD)**: A Media Presentation Description (MPD) file is created. This is an XML document that acts as a playlist for the content. The MPD contains essential information, including:
   - Metadata about the content, such as title and description.
   - A list of available streams, along with their characteristics (bitrate, resolution, codecs).
   - URLs to the media segments for each stream.
3. **Server Hosting**: The content, along with the MPD file, is hosted on a web server or a content delivery network (CDN). The server is responsible for serving these files to clients on request.
4. **Client Request**: When a viewer initiates playback, the client (media player or application) sends an HTTP GET request to the server for the MPD file. This is the initial request.

5. **MPD Parsing**:The client parses the MPD to determine the available streams and their characteristics. Based on the viewer's network conditions, device capabilities, and preferences, the client selects an appropriate stream to begin playback.
6. **Segment Retrieval**: The client, based on its selected stream, starts sending HTTP GET requests for the individual media segments to the server. These segments are downloaded and cached locally for playback.
7. **Adaptive Bitrate Streaming:** The client continually monitors network conditions and viewer preferences. If conditions change (e.g., available bandwidth increases or decreases), the client may switch to a different quality level by requesting segments from another stream with an appropriate bitrate.

8. **Playback and Continuous Retrieval**:The client plays back the content as segments are retrieved and cached. It continuously requests new segments as playback progresses to ensure a seamless viewing experience.
MPEG-DASH enables adaptive streaming by allowing the client to switch between different quality levels based on network conditions. It leverages standard HTTP for content delivery, making it compatible with web servers and CDNs. Its flexibility, broad device compatibility, and standardization have made it a popular choice for multimedia content delivery over the internet.[12]

3.7 Low-Latency CMAF with MPEG-DASH

Low latency CMAF (Common Media Application Format) with MPEG-DASH (Dynamic Adaptive Streaming over HTTP) combines the principles of low latency streaming with the flexibility and compatibility of the MPEG-DASH streaming protocol. It is designed to reduce the latency in delivering multimedia content over the internet using the CMAF format and the MPEG-DASH protocol.

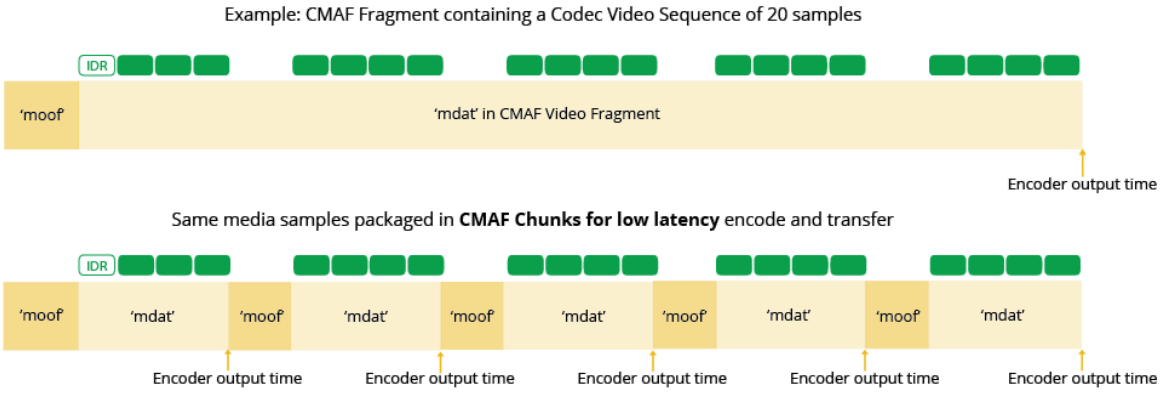

Fig 3.6.1 moof(movie fragments) and mdat(media data) or fragments within a CMAF file

The "**mdat**" contains the raw media data samples, while "**moof**" serves as a packaging structure for organizing media samples and associated metadata within specific time intervals. Together, they facilitate the efficient delivery and playback of multimedia content in CMAF, especially in adaptive streaming scenarios where content is segmented and delivered at different bitrates and resolutions.

CMAF for Low Latency:
CMAF is a key component of low latency streaming. CMAF defines a media format that allows content to be divided into smaller, more frequent segments or "chunks." These chunks are typically a

few seconds or less in duration. This format is well-suited for low latency streaming because it enables quicker delivery of content. In the context of low latency CMAF with MPEG-DASH, the content encoding, packaging, and delivery processes are optimized for parallel processing. This means that the content is encoded and packaged as close to real-time as possible. The server hosting the content is designed to efficiently deliver these smaller, more frequent chunks in real-time.
This combination is well-suited for applications where real-time or near-real-time interactions with viewers are critical. Examples include live sports broadcasts, online gaming, and interactive streaming, where minimizing the delay between content capture and viewer playback is essential.[06]

3.8 Comparative Study of Streaming Protocols

Several protocols were widely used for audio and video streaming, each with its unique characteristics. HTTP Adaptive Streaming Protocols, such as HLS and MPEG-DASH, are popular for their broad compatibility and adaptive streaming capabilities, making them efficient choices for delivering content to a diverse range of devices. Real-Time Streaming Protocols like RTMP and WebRTC have been prominent for real-time and interactive applications, with WebRTC excelling in low-latency scenarios. Low Latency CMAF, in combination with MPEG-DASH or HLS, is gaining traction for applications demanding ultra-low latency. While the choice of protocol depends on specific use cases, considerations include latency requirements, adaptive streaming capabilities, device compatibility, and content security.

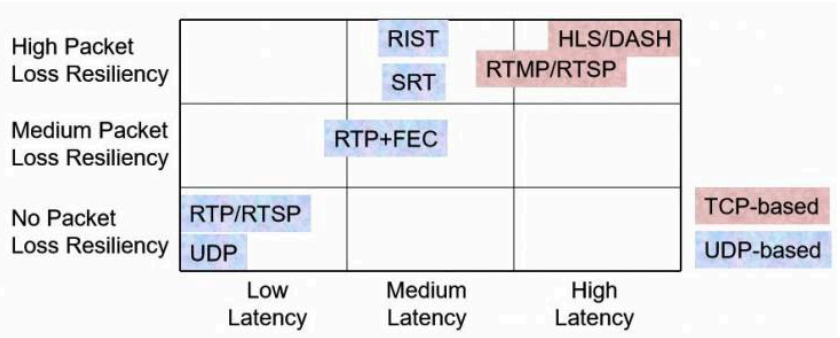

Fig 3.7.1 Comparision of Protocols

Multimedia Streaming protocols can be compared in the following areas:

1. First-mile contribution vs. last-mile delivery
2. Encoder support
3. Latency
4. Quality of experience (adaptive bitrate enabled, etc.)
5. Security
6. Scalability
7. Playback support

Choosing the appropriate media streaming protocol begins by clarifying your specific objectives. Factors like latency, compatibility with playback devices, and the overall quality of the viewing experience can all come into play. It's important to note that content distributors often employ different protocols at various stages, not adhering to a single protocol from content capture to playback. For instance, many broadcasters employ RTMP for the initial transfer from the encoder to the server and subsequently convert the stream into an adaptive HTTP-based format.

4. Video and Audio Compression

Video and audio compression are essential in video streaming to enhance transmission efficiency. Compression techniques reduce redundancy among consecutive frames, employing coding methods to condense large, raw video streams, rendering them more suitable for transmission across networks. By selectively reducing quality, compression reduces both file size and bitrate. This reduction in file size and bitrate translates to a need for less bandwidth when transmitting compressed videos over the internet. As a result, streaming becomes smoother, buffering is minimized, and performance improves on a variety of devices, even those with slower internet connections.

4.1 Codec Selection

Video compression starts with selecting the appropriate codec (compression-decompression algorithm). Common video codecs include H.264 (also known as AVC), H.265 (HEVC), VP9, and AV1. These codecs use various techniques to reduce the size of video data while maintaining acceptable visual quality.

4.2. Lossy vs. Lossless Compression

Most video codecs employ lossy compression, which means they discard some visual data to achieve higher compression ratios. Lossless compression retains all the original data but typically results in larger file sizes. In streaming, lossy compression is generally used to save bandwidth.

4.2.1 Lossy Compression

Lossy compression algorithms achieve data reduction by eliminating portions of the original data that the human sensory system is less likely to notice. This often includes fine details, subtle color variations, or inaudible audio frequencies.This algorithm is irreversible, meaning that once data is compressed and portions are discarded, it cannot be perfectly reconstructed to its original state. This is in contrast to lossless compression, which allows for perfect reconstruction.Lossy compression techniques are designed with human perception in mind. They exploit characteristics of human vision and hearing to remove data that is unlikely to be detected by the human senses. For example, in image compression, the removal of fine image details that are not easily discernible to the human eye.[07]

Lossy audio codecs like MP3 and AAC (Advanced Audio Coding) are widely used for music and audio streaming. They remove imperceptible audio data to create smaller file sizes while maintaining good audio quality. Lossy video compression standards, like H.264 and H.265 (HEVC), are extensively used in video streaming, video conferencing, and digital television. They efficiently reduce the size of video files while retaining reasonable visual quality.

4.2.2 Lossless Compression

Unlike lossy compression, which sacrifices some data to achieve smaller file sizes, lossless compression retains all the original data. This ensures that when the compressed data is decompressed, it's an exact replica of the original. Lossless compression is reversible, meaning it can perfectly reconstruct the original data. It's commonly used in applications where the integrity of the data is crucial, such as archiving, text document storage, and data transmission.Lossless compression

algorithms are typically general-purpose and can be applied to various types of data, including text, images, audio, and more.
Lossless compression is widely used for text and document compression formats, like ZIP, which reduces the size of text files, spreadsheets, and other documents without losing any data. Some audio formats, like FLAC (Free Lossless Audio Codec), use lossless compression to maintain audio quality while reducing file sizes.

Aspect	Lossy Compression	Lossless Compression
Data Loss	Some data is lost during compression	No data loss; data is preserved intact
Reversibility	Irreversible; cannot perfectly reconstruct the original data	Reversible; the original data can be perfectly reconstructed
Compression Ratio	Achieves higher compression ratios	Achieves lower compression ratios
Use Cases	Multimedia (images, audio, video) where some loss of quality is acceptable	Text, documents, archiving, where data integrity is critical
Audio Formats	MP3, AAC, and others	FLAC, ALAC, and others
Storage and Archiving	Less suitable for archiving due to data loss	Ideal for archiving and data backup
Computational Resources	Generally less resource-intensive	May be more resource-intensive, especially for complex data

Table 4.2.1 Lossy vs Lossless compression

4.3. Bit Rate Control

Streaming protocols often adapt the bit rate of the video stream based on network conditions. Adaptive streaming protocols, like HLS and MPEG-DASH, allow the server to send video at varying bit rates, adjusting to the viewer's available bandwidth. This is especially important for ensuring smooth playback.
Constant Bit Rate (CBR) **vs**. **Variable Bit Rate** (VBR): Streaming protocols may use either CBR or VBR encoding for audio. CBR maintains a consistent bit rate but can be less efficient, while VBR adjusts the bit rate as needed, optimizing for audio quality.

In both video and audio compression, the goal is to strike a balance between reducing data size and maintaining acceptable quality. The use of compression techniques, codecs, and adaptive streaming ensures that streaming protocols can deliver content efficiently while adapting to the viewer's network conditions and device capabilities. The choice of specific compression settings and codecs may vary depending on the content, target audience, and the specific requirements of the streaming platform.

4.4 Video and Audio Compression Standards

Here's a table that lists common audio and video compression standards used in streaming technologies:

Compression Standard	Type	Description	Streaming Protocols
H.264/AVC	Video	Provides efficient video compression with good quality.	RTMP, HLS, MPEG-DASH, WebRTC
H.265/HEVC	Video	Offers high compression efficiency, suitable for high-definition and ultra-high-definition content.	HEVC over HTTP, MPEG-DASH
VP9	Video	An open-source video codec with efficient compression.	WebM, WebRTC
AV1	Video	Open-source codec designed for high-quality compression.	V1 over HTTP, MPEG-DASH
MP3 (MPEG-1 Audio Layer III)	Audio	Widely used audio compression for music and audio playback.	HTTP, RTSP, HLS, MPEG-DASH
AAC (Advanced Audio Coding)	Audio	Provides improved audio quality over MP3.	HTTP, RTSP, HLS, MPEG-DASH
Opus	Audio	Open-source codec for low-latency audio streaming and voice communication.	WebRTC, SIP, VoIP protocols
FLAC (Free Lossless Audio Codec)	Audio	A lossless compression format preserving original audio quality.	HTTP, FLAC streaming protocols

Table 4.4.1 Common Compression Standards used by Streaming Protocols

These are some of the common standards used in streaming technologies. The choice of compression standard depends on factors such as the content's nature, desired quality, available bandwidth, and device compatibility. Keep in mind that there are many other codecs and standards available, each with its unique features and applications.

5. Content Delivery Network

A Content Delivery Network (CDN) is a network of distributed servers strategically located in various data centres and points of presence (PoPs) around the world. The primary purpose of a CDN is to deliver web content, such as text, images, videos, and other resources, to users quickly and efficiently, while also enhancing the overall performance, security, and availability of web services.
A CDN caches and stores copies of web content, including images, videos, scripts, stylesheets, and more, on its servers located in different geographic regions. This allows users to access content from a server that is physically closer to them, reducing latency and load times. CDNs use various optimization techniques, such as data compression, minification, and content prefetching, to accelerate the delivery of web pages. This results in faster loading times for websites and web applications.
CDNs help distribute incoming traffic across multiple servers to prevent any single server from becoming overwhelmed. This load balancing ensures high availability and reduces the risk of server crashes or downtime.Here are key features and functions of a CDN:

1. **DDoS Protection**: CDNs often include DDoS (Distributed Denial of Service) protection services. By distributing traffic across multiple servers and implementing security measures, CDNs can help mitigate DDoS attacks and maintain the availability of web services.
2. **Scalability**: CDNs provide a scalable infrastructure that can handle spikes in traffic, which is particularly valuable for websites and applications experiencing rapid growth or sudden increases in demand.
3. **Security**: CDNs offer security features, such as SSL/TLS encryption, web application firewall (WAF), and protection against various types of web threats and attacks.
4. **Global Reach**: CDNs have a global network of servers, enabling websites and content to be delivered to users worldwide with minimal latency.
5. **Analytics and Reporting**: CDNs often provide analytics and reporting tools that allow website owners to monitor and analyze traffic patterns, performance, and user behaviour.
6. **Video and Streaming**: CDNs are crucial for delivering video content and live streaming events, as they can handle the large bandwidth requirements and ensure smooth playback.
7. **Mobile Optimization**: CDNs are optimized for mobile delivery, helping to deliver content efficiently to mobile devices, which often have slower or less reliable network connections.

Widespread CDN providers comprise of Amazon CloudFront, Akamai, Cloudflare, and Fastly. Website owners and businesses typically integrate a CDN with their web hosting to improve the performance and availability of their online content, ensuring a better user experience for visitors from different locations and devices.[08]

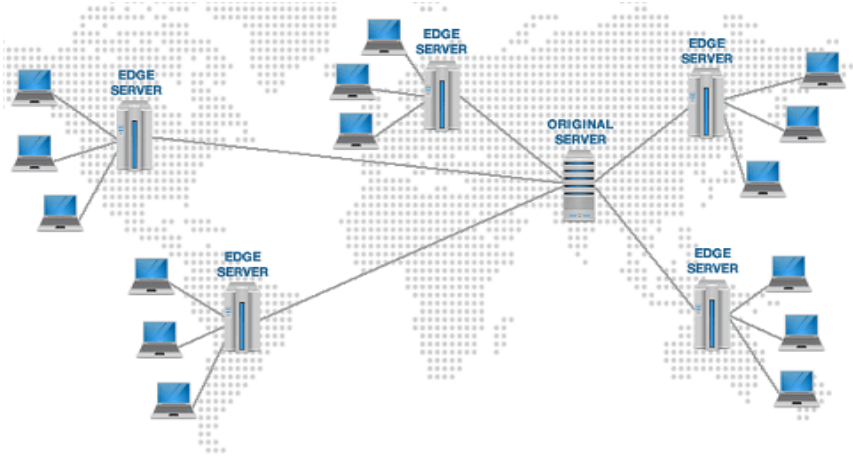

Fig 5.1 Content Delivery Network Edge Locations

6. Future Trends

Future trends in streaming technologies from the perspective of computer networks are likely to be driven by advancements in technology and changing user expectations. Here are some of the emerging trends in streaming technologies and how they relate to computer networks:

1. **Multi-Platform and Cross-Device Streaming:** Streaming services are becoming device-agnostic, allowing users to seamlessly switch between different devices while enjoying consistent content access. This trend will continue to grow with the proliferation of smartphones, smart TVs, and connected devices.
2. **Immersive Streaming:** Virtual reality (VR) and augmented reality (AR) streaming will become more prevalent. These technologies will demand higher network bandwidth and lower latency to provide immersive experiences, making edge computing and high-speed networks crucial.[9]

3. **Dynamic Adaptive Streaming over QUIC (DASQ)**: DASQ is an emerging technology that combines the benefits of Dynamic Adaptive Streaming (DASH) with the speed and security of the QUIC (Quick UDP Internet Connections) protocol. This offers faster and more secure streaming experiences, especially when streaming over unreliable networks.
4. **Multi-CDN Strategies**: Businesses will increasingly adopt multi-CDN strategies to ensure redundancy, minimize downtime, and improve global content delivery. This approach optimizes performance by using multiple CDN providers.
5. **Content Protection and DRM**: As streaming content becomes more valuable, ensuring content protection through robust digital rights management (DRM) systems and anti-piracy measures will remain a priority.
6. **Adaptive Network Protocols**: Protocols like HTTP/3 and QUIC will continue to evolve to improve streaming performance, reliability, and security.
7. **Multi-Channel Networks (MCNs):** Platforms will continue to partner with MCNs to provide a wide range of content from independent creators, diversifying their content libraries.

These trends underscore the importance of robust computer networks to meet the growing demands of streaming technologies. As more high-quality content is delivered over the internet, network infrastructure, including high-speed connectivity, security measures, and efficient content delivery systems, will play a crucial role in shaping the future of streaming.[10]

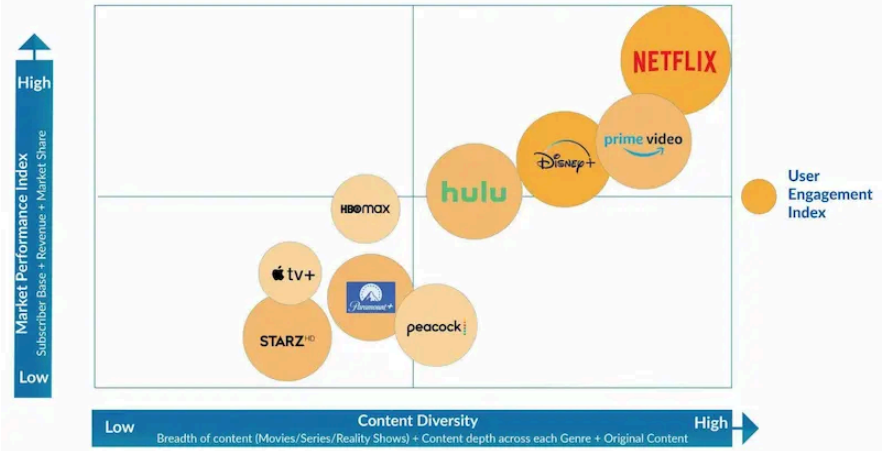

Fig 6.1 Growing Market Performance of Streaming Platforms

7. Conclusion

In conclusion, this research paper has provided an in-depth exploration of the fundamentals of streaming, various traditional and latest streaming protocols, audio and video compression, and the future trends shaping the streaming landscape. We have delved into the intricacies of key streaming protocols such as RTMP, RTSP, and HLS, understanding their architectures and applications in the context of computer networks. We have also examined the essential role of audio and video compression, differentiating between lossy and lossless compression, and how they optimize data transmission for streaming.

As we have observed, the streaming industry is continually evolving, and future trends are poised to bring about significant changes. With the rise of 5G networks, edge computing, and low-latency technologies, the streaming experience is becoming more immersive and interactive. The adoption of adaptive network protocols and the integration of AI and machine learning for content

recommendation and quality enhancement will continue to shape the future of streaming. Furthermore, the expansion of streaming platforms, user-generated content, and the incorporation of AR, VR, and blockchain technologies offer exciting possibilities for the streaming ecosystem.

In this dynamic environment, it is essential for stakeholders in the streaming industry to stay up-to-date on these developments, ensuring that they harness the full potential of emerging technologies while addressing challenges like content protection and sustainability. As content creators, platform providers, and consumers, we are witnessing a revolution in how we access and enjoy audiovisual content. The future of streaming promises more personalized, interactive, and high-quality experiences, making it an exciting era for both technology enthusiasts and content aficionados.

In summary, this research paper has unveiled the intricate workings of streaming protocols and compression techniques while offering a glimpse into the exciting future trends that will continue to shape the ever-evolving world of streaming. It is our hope that this paper has provided valuable insights into this fascinating and rapidly advancing field, inspiring further research and innovation in the realm of streaming technologies.

8. Acronyms

Acronym	**Full Form**
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VOD	Video On Demand
OTT	Over The Top
RTMP	Real Time Messaging Protocol
RTMPS	Real Time Messaging Protocol Secured
RTMPT	Real Time Messaging Protocol Tunneled
RTMFP	Secure Real-Time Media Flow Protocol
TLS/SSL	Transport Layer Security/ Secure Socket Layer
RTSP	Real Time Streaming Protocol
RTP	Real Time Protocol
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
MPEG-DASH	Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP
CMAF	Common Media Application Format

CDN	Content Delivery Network
DASQ	Dynamic Adaptive Streaming over QUIC
QUIC	Quick UDP Internet Connections
MCN	Multi-Channel Networks
DRM	Digital Rights Management
VR	Virtual Reality
AR	Augmented Reality
CBR	Constant Bit Rate
VBR	Variable Bit Rate
AVC	Advanced Video Coding
HVEC	High-Efficiency Video Coding
DDoS	Distributed Denial of Service
PoP	Point of Presence
MPD	Media Presentation Description
MP3	MPEG-1 Audio Layer III
AAC	Advanced Audio Coding
FLAC	Free Lossless Audio Codec

9. References

01. Chin-Feng Lai, Han-Chieh Chao, Ying-Xun Lai, Jiafu Wan, "Cloud-assisted real-time transrating for http live streaming", IEEE Wireless Communications, vol.20, no.3, pp.62-70, 2013.

02. P. Frossard, J. C. de Martin and M. R. Civanlar, "Media Streaming With Network Diversity," in Proceedings of the IEEE, vol. 96, no. 1, pp. 39-53, Jan. 2008

03. A. Dutta and H. Schulzrinne, "A streaming architecture for next generation Internet," ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240), Helsinki, Finland, 2001

04. W. Xin-Gong, "Research on Multimedia Data Stream in Network," 2013 Fourth International Conference on Intelligent Systems Design and Engineering Applications, Zhangjiajie, China, 2013

05. G. Lu, “Issues and technologies for supporting multimedia communications over the Internet” 2000 The International Journal for the Computer and Telecommunications Industry, Torino, Italy, 2000

06. David Austerberry. 2005. The Technology of Video and Audio Streaming Second Edition. Focal Press: An imprint of Elsevier.

07. John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee, “Video Streaming: Concepts, Algorithms, and Systems,” 2002 Mobile and Media Systems Laboratory HP Laboratories, Palo Alto, California, USA, 2002

08. J. Summers, T. Brecht, D. Eager and A. Gutarin, "Characterizing the workload of a netflix streaming video server," 2016 IEEE International Symposium on Workload Characterization (IISWC), Providence, RI, USA, 2016

09. Ho, K.M., Lo, K.T., Feng, J. (2006). Multimedia Streaming on the Internet. In: Furht, B. (eds) Encyclopedia of Multimedia. Springer, Boston, MA. https://doi.org/10.1007/0-387-30038-4_166

10. Cisco. (2018) Cisco Best Practices Streaming Video for High-Impact Business Communications https://www.cisco.com/c/dam/en_us/about/ciscoitatwork/downloads/ciscoitatwork/pdf/cisco_streaming_white_paper.pdf

11. H. Schulzrinne,Columbia U.,A. Rao. (2023). Real Time Streaming Protocol (RTSP) https://www.rfc-editor.org/rfc/rfc2326.html

12. MPEG.org (2012). MPEG-DASH. https://www.mpeg.org/standards/MPEG-DASH/

13. Apple Inc. (2023).HTTP Live Streaming. https://developer.apple.com/documentation/http-live-streaming

14. Max Wilbert. (2023). Video Streaming Protocols: 6 Preferred Formats for Professional Broadcasting. https://www.dacast.com/blog/video-streaming-protocol/

15. Christopher Mueller. (2022). MPEG-DASH (Dynamic Adaptive Streaming over HTTP) https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/

16. Traci Ruether. (2022). Streaming Protocols: Everything You Need to Know (Update). https://www.wowza.com/blog/streaming-protocols