applied sciences

MDPI

Article

A Decentralized Multi-Venue Real-Time Video Broadcasting System Integrating Chain Topology and Intelligent Self-Healing Mechanisms

Tianpei Guo [1], Ziwen Song [2], Haotian Xin [3] and Guoyang Liu [1,*]

[1] School of Integrated Circuits, Shandong University, Jinan 250101, China; 202200201042@mail.sdu.edu.cn
[2] School of Software, Shandong University, Jinan 250101, China; songzw@mail.sdu.edu.cn
[3] School of Political Science and Public Administration (SPSPA), Shandong University, Qingdao 266237, China; 202300120329@mail.sdu.edu.cn
* Correspondence: gyliu@sdu.edu.cn

Abstract

The rapid growth in large-scale distributed video conferencing, remote education, and real-time broadcasting poses significant challenges to traditional centralized streaming systems, particularly regarding scalability, cost, and reliability under high concurrency. Centralized approaches often encounter bottlenecks, increased bandwidth expenses, and diminished fault tolerance. This paper proposes a novel decentralized real-time broadcasting system employing a peer-to-peer (P2P) chain topology based on IPv6 networking and the Secure Reliable Transport (SRT) protocol. By exploiting the global addressing capability of IPv6, our solution simplifies direct node interconnections, effectively eliminating complexities associated with Network Address Translation (NAT). Furthermore, we introduce an innovative chain-relay transmission method combined with distributed node management strategies, substantially reducing reliance on central servers and minimizing deployment complexity. Leveraging SRT's low-latency UDP transmission, packet retransmission, congestion control, and AES-128/256 encryption, the proposed system ensures robust security and high video stream quality across wide-area networks. Additionally, a WebSocket-based real-time fault detection algorithm coupled with a rapid fallback self-healing mechanism is developed, enabling millisecond-level fault detection and swift restoration of disrupted links. Extensive performance evaluations using Video Multi-Resolution Fidelity (VMRF) metrics across geographically diverse and heterogeneous environments confirm significant performance gains. Specifically, our approach achieves substantial improvements in latency, video quality stability, and fault tolerance over existing P2P methods, along with over tenfold enhancements in frame rates compared with conventional RTMP-based solutions, thereby demonstrating its efficacy, scalability, and cost-effectiveness for real-time video streaming applications.

Keywords: decentralized video broadcasting; real-time streaming; SRT protocol; IPv6 network; chain topology; intelligent self-healing; fault tolerance; low-latency transmission; distributed systems; peer-to-peer architecture

Academic Editor: Christos Bouras

Received: 21 June 2025
Revised: 14 July 2025
Accepted: 16 July 2025
Published: 19 July 2025

Citation: Guo, T.; Song, Z.; Xin, H.; Liu, G. A Decentralized Multi-Venue Real-Time Video Broadcasting System Integrating Chain Topology and Intelligent Self-Healing Mechanisms. *Appl. Sci.* **2025**, *15*, 8043. https://doi.org/10.3390/app15148043

1. Introduction

Distributed video streaming systems have undergone a significant technological evolution over the past decade, shifting from centralized to increasingly decentralized paradigms [1]. Traditional streaming architectures face considerable technical bottlenecks

in scenarios involving large-scale concurrency and geographic distribution [2]. The fundamental limitations of centralized architectures stem from their inherent single point of failure risks and linear scaling costs, as server bandwidth and computational resource demands escalate exponentially with user growth [3,4].

As the scale of streaming applications continues to expand, the traditional centralized server-centric content-distribution model has revealed clear scalability bottlenecks [5]. Centralized architectures typically suffer from soaring server loads, network-bandwidth saturation, and single points of failure during high-concurrent access [6]. Concurrently, the upload bandwidth and computing resources of edge nodes often remain underutilized. Peer-to-peer (P2P) network architectures address the inherent scalability issues of centralized systems by enabling each node in the network to act as both a consumer and a distributor of content, thus achieving a dynamic balance between resource supply and demand [2]. In P2P streaming systems, nodes can directly establish data transmission links, forming distributed content distribution networks, such as chain or mesh topologies. This effectively offloads the central server, enhancing system resilience and fault tolerance [7]. However, large-scale P2P deployments introduce their own set of technical challenges, including topology instability due to dynamic node behavior (frequent online/offline transitions), maintaining content data consistency, multi-source synchronization control, and mechanisms for identifying and protecting against malicious nodes [8]. These factors impose stringent requirements on system real-time performance and robustness.

However, existing P2P streaming solutions, while alleviating the load on central servers, still face several critical limitations. First, topological instability is a persistent challenge: frequent node churn (nodes joining and leaving) leads to dynamic changes in the overlay network, which can cause link disruptions and degrade service continuity. Second, synchronization across heterogeneous networks and devices is difficult to guarantee, resulting in inconsistent playback quality and increased latency, especially in multi-source or multi-hop scenarios. Third, many P2P systems lack robust mechanisms for rapid fault detection and recovery, making them vulnerable to cascading failures. Moreover, security and trust management remain open issues, as malicious or low-quality nodes can disrupt the overall service. Additionally, the scarcity of IPv4 addresses has necessitated widespread deployment of Network Address Translation (NAT) technologies, which significantly complicate P2P implementations. NAT creates complex network topologies where direct peer-to-peer connections become challenging or impossible, often requiring sophisticated traversal techniques such as STUN, TURN, or UPnP that introduce additional performance overhead and connection establishment delays [9,10]. These NAT-related complexities not only increase system deployment difficulty but also reduce connection reliability and introduce unpredictable latency variations that can severely impact real-time streaming performance. These limitations highlight the need for more resilient, adaptive, and self-healing P2P architectures.

In recent years, real-time communication scenarios such as remote conferencing, online education, and interactive live streaming have posed increasingly stringent requirements on video transmission, especially regarding latency, jitter, and data integrity [11]. While traditional TCP-based streaming protocols (e.g., RTMP, HTTP-FLV, or M3U8-MPEGTS) [12–14] offer sufficient data reliability, the acknowledgment-retransmission mechanisms and congestion control strategies brought by the TCP protocol introduce significant jitter during link congestion or packet loss. This can disrupt audio–video synchronization, particularly in weak network conditions or across international Wide Area Networks (WANs) [15].

To address these challenges, several transport protocols have been proposed. The Secure Reliable Transport (SRT) protocol, an emerging reliable transport protocol built over UDP, ensures low-latency data transmission while providing mechanisms like Selective

Repeat ARQ, dynamic window adjustment, and sophisticated congestion control [16]. SRT also integrates end-to-end AES encryption, ensuring the security of transmitted content [17]. In contrast, WebRTC and QUIC have emerged as alternative solutions with their own strengths and limitations. WebRTC, designed primarily for browser-based peer-to-peer communication, offers built-in NAT traversal via ICE, STUN, and TURN protocols, along with comprehensive media capabilities including adaptive codecs and bandwidth estimation [18]. However, WebRTC's complex signaling requirements, dependency on centralized servers for session establishment, and challenges in scaling beyond small mesh topologies limit its suitability for large-scale broadcasting architectures. QUIC, initially developed by Google and now standardized as HTTP/3, provides multiplexed connections over UDP with improved connection establishment, better congestion control, and built-in encryption [19]. While QUIC offers benefits like connection migration and reduced head-of-line blocking, its optimization for HTTP traffic rather than real-time media, inconsistent implementation across platforms, and relatively higher computational overhead present challenges for streaming applications [20]. Compared with these alternatives, SRT is specifically optimized for professional media streaming scenarios, offering mature support for multi-hop relays, fine-grained retransmission mechanisms, and robust encryption, which are critical for decentralized chain-based architectures.

The widespread adoption of P2P architectures has historically been constrained by Network Address Translation (NAT) mechanisms [21]. Due to the scarcity of IPv4 addresses, most terminal devices reside in private address spaces, making direct inter-node communication challenging. This necessitated reliance on complex NAT traversal techniques (e.g., STUN, TURN, or UPnP) [9,10,22] for connection establishment, increasing system development and deployment complexity. With the global deployment of the IPv6 protocol, the internet infrastructure is undergoing a fundamental shift from "address reuse" to "end-to-end reachability" [23]. IPv6 assigns each endpoint a unique global address, enabling direct end-to-end connectivity while significantly reducing the signaling overhead and latency typically introduced by relay-based P2P architectures [24]. More importantly, IPv6's larger address space, improved routing, and built-in security offer a strong foundation for scalable real-time streaming, making decentralized systems easier to deploy and more efficient to operate. Multi-venue real-time video broadcasting represents a typical distributed streaming application that imposes strict requirements on system real-time performance, reliability, and scalability. Existing solutions primarily fall into two categories: centralized distribution and P2P-based collaborative transmission [25]. While centralized solutions offer excellent Quality of Service (QoS) assurance, they are often characterized by high deployment costs and limited scalability [26]. P2P solutions, despite their scalability, still face challenges in node stability and service continuity [27]. Recent efforts in hybrid P2P-CDN systems have demonstrated that integrating peer-based transmission with adaptive delivery and edge-assisted computing can reduce latency and improve user experience in live video streaming [28]. However, these hybrid approaches often rely on centralized coordination or edge servers, which reintroduce single points of failure and limit true decentralization. Moreover, recent studies (e.g., [7,29,30]) have shown that even advanced P2P and hybrid systems may suffer from recovery delays exceeding 3–5 s and packet loss spikes above 1% under adverse conditions, whereas our proposed system achieves sub-second recovery and maintains packet loss below 0.5% in multi-hop scenarios. This quantifiable performance gap underscores the necessity for more robust, self-healing architectures.

In summary, the proposed system distinguishes itself from recent works by introducing a fully decentralized, chain-based relay architecture with intelligent self-healing mechanisms. Unlike prior solutions, our approach enables rapid fault detection and au-

tonomous relay path reconstruction, minimizing service interruption and ensuring stable video quality even in highly dynamic or adverse network environments. The following sections detail the system design, experimental methodology, results, and discussion, providing a comprehensive analysis of the proposed approach and its advantages over existing methods.

Based on these premises, this paper proposes a decentralized chain-topology-based multi-venue real-time broadcasting system that integrates the SRT protocol, IPv6 networks, and intelligent fault recovery mechanisms. The main contributions of the paper are as follows:

- We propose an IPv6-enabled direct-connected P2P transmission architecture, significantly reducing bandwidth and hardware performance overhead of traditional centralized streaming solutions, avoiding complexities associated with traditional NAT traversal methods, and enabling streamlined connectivity and efficient network scalability, as illustrated in Figure 1. This architecture allows for direct inter-node communication, eliminating the need for complex NAT traversal techniques and enabling efficient resource utilization across distributed nodes.

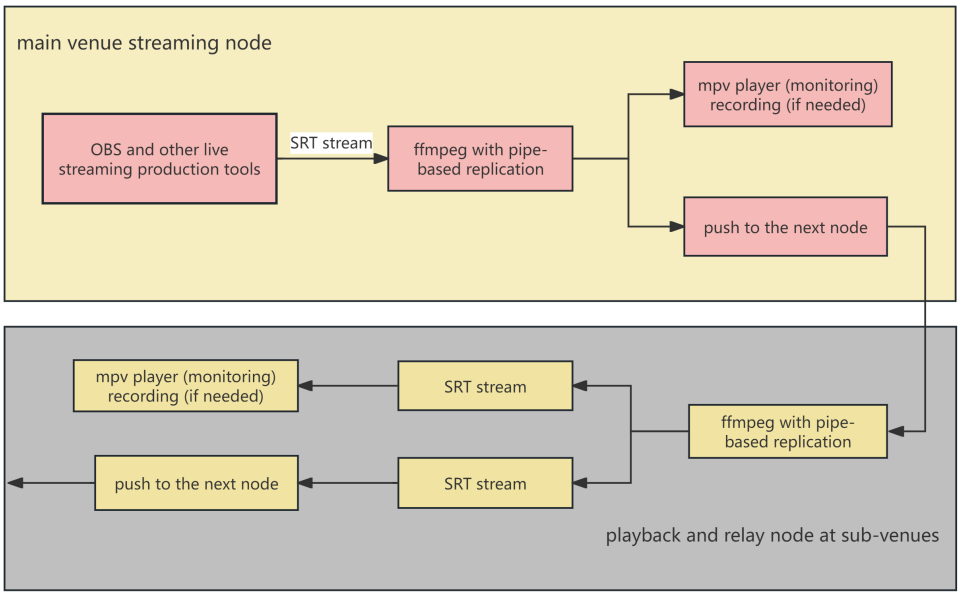


Figure 1. System architecture overview.

- Leveraging the Secure Reliable Transport (SRT) protocol, we achieve low-latency, high-reliability video streaming. SRT's selective retransmission, congestion-aware control, and AES-128/256 encryption mechanisms collectively ensure secure and resilient data transport.
- We introduce a WebSocket-driven real-time fault detection and rapid self-healing mechanism capable of detecting faults at millisecond granularity and restoring network links within seconds, substantially improving system robustness.
- Extensive evaluations conducted under diverse network conditions and hardware configurations demonstrate the superior performance of the proposed system, including minimal end-to-end latency, sustained high video quality, and exceptional stability and reliability against link disruptions.

Our system strategically employs IPv6 direct-connected P2P transmission topology, effectively eliminating the complexities of traditional NAT traversal. We further integrate SRT protocol's reliable transport mechanisms, which, leveraging UDP's speed, ensure low latency, robust data transfer through selective retransmission and advanced congestion

control, and provide end-to-end security via AES-128/256 encryption. To specifically enhance resilience, we introduce a WebSocket-based real-time fault detection algorithm and a fallback self-healing mechanism, achieving millisecond-level fault discovery and second-level link reconstruction. Through comprehensive performance evaluations, we validate the effectiveness of the system across various network environments and hardware platforms, demonstrating its superior end-to-end latency, video quality stability, and link fault tolerance. This system is applicable in scenarios such as inter-campus academic conferences, multi-location remote education, smart city emergency command, and cross-border enterprise live events. In such applications, direct IPv6-based node interconnection eliminates NAT traversal, while SRT ensures low-latency and reliable data delivery, making the system highly suitable for deployment in bandwidth-constrained or dynamically changing environments.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of related work in P2P video streaming, including P2P-assisted HTTP streaming, decentralized crowdsourced systems, and high-quality P2P streaming architectures. Section 3 presents the detailed system design, covering the IPv6-based P2P architecture, SRT protocol integration, chain topology construction, intelligent self-healing mechanisms, and deployment implementation details. Section 4 describes the experimental methodology and presents extensive performance evaluations, including benchmarks, comparative analysis, and fault tolerance assessments across diverse network environments and hardware configurations. Section 5 presents the experimental results and performance analysis. Section 6 discusses the implications of our findings, limitations of the current approach, and potential future research directions. Finally, Section 7 concludes the paper and summarizes the key contributions.

2. Related Work

2.1. P2P-Assisted HTTP Video Streaming

P2P-assisted HTTP streaming technologies integrate traditional client–server architectures with P2P collaborative mechanisms to effectively alleviate the load on central servers. Such systems typically employ hybrid architectures where servers handle initial content distribution and node coordination, while the P2P network manages horizontal content propagation and load balancing [28]. Building upon this, the SmoothCache system [31] represents an early and influential work in P2P-assisted HTTP streaming. Specifically designed for HTTP Live Streaming (HLS) protocol characteristics, SmoothCache developed a dynamic transmission scheduling algorithm based on segment urgency. Its core technical innovations include an application-layer congestion control mechanism that dynamically adjusts transmission priorities according to network conditions, an intelligent neighbor selection algorithm that identifies optimal upload nodes based on Round-Trip Time (RTT) and available bandwidth, and a proactive caching strategy that anticipates user demand and pre-caches popular content. Experimental results confirmed that SmoothCache significantly reduced server bandwidth consumption while maintaining a positive user playback experience in high-concurrency scenarios.

Hybrid P2P-CDN architectures have further expanded upon this concept [32]. The ALIVE system, proposed by Farahani et al. [32], integrated Network Function Virtualization (NFV) and edge computing technologies to achieve dynamic Quality of Experience (QoE) optimization. Key innovations in ALIVE included a machine learning-based node matching algorithm capable of dynamically selecting optimal transmission paths based on network conditions and user preferences [33], a distributed transcoding mechanism that performed real-time transcoding at edge nodes to adapt to diverse device requirements, and multi-tenant resource scheduling through virtualization for efficient resource sharing.

More recent research, exemplified by the RICHTER system [34], further introduced Self-Organizing Map (SOM) algorithms and online learning mechanisms to enable intelligent resource matching under large-scale concurrent requests. This system employed a collective optimization strategy, clustering user requests with similar characteristics to significantly enhance scheduling efficiency and resource utilization. Despite these advancements, existing P2P-assisted HTTP streaming systems still exhibit notable limitations. These include their continued reliance on central coordination nodes, which introduces single points of failure, inherent real-time limitations due to the HTTP-based transport protocol, and limited capabilities for optimizing latency in long-distance, cross-regional transmissions [35].

2.2. Decentralized Crowdsourced Video Streaming Systems

Decentralized crowdsourced video streaming systems represent a crucial application of P2P technology in the User-Generated Content (UGC) domain. Unlike traditional content delivery networks, crowdsourced systems face unique technical challenges such as high content source heterogeneity, inconsistent data quality, and complex real-time synchronization [30]. At the commercial level, major live streaming platforms like Twitch and YouTube Live, while providing user-friendly service interfaces, fundamentally rely on large-scale cloud computing infrastructures, leading to high operational costs and inherent centralization risks. In recent years, blockchain-based decentralized streaming platforms such as Livepeer [26] and Theta Network have begun exploring truly decentralized solutions, although these nascent systems still face technical bottlenecks concerning real-time performance and scalability. In academic research, Geng and Fujita [30] proposed a distributed multi-source video acquisition and distribution system specifically tailored for large-scale events. Its technical architecture comprised node discovery and routing mechanisms based on the Kademlia Distributed Hash Table (DHT), a Bitswap protocol-driven content exchange algorithm for efficient video segment distribution, and an FFmpeg-driven multi-source video synthesis engine supporting real-time multi-view stitching. Experimental results demonstrated the system's effectiveness in handling multiple concurrent video streams while providing a positive user experience.

WebRTC technology offers an alternative and increasingly viable technical pathway for decentralized video transmission [18]. Research by Diallo et al. indicates that WebRTC-based pure P2P video systems can operate stably on resource-constrained embedded devices. Their key advantages include direct end-to-end communication without server intermediation, built-in NAT traversal mechanisms that simplify network configuration, and adaptive encoding technology ensuring transmission quality across varying network conditions. Performance evaluations of such systems typically show excellent CPU utilization, memory consumption, and bandwidth efficiency.

Beyond core transmission, reputation management mechanisms play a vital role in enhancing the stability and efficiency of decentralized systems. The ReputeStream system [7] effectively mitigates node free-riding behavior by introducing a Bayesian theory-based reputation evaluation algorithm. This system employs a multi-layer overlay architecture, strategically deploying high-reputation nodes closer to content sources, which significantly improves the overall network's transmission efficiency and stability [29]. Despite substantial advancements in decentralized crowdsourced video systems, particularly with innovative solutions for reputation mechanisms, node collaboration, and content scheduling [29], several key challenges persist in practical large-scale deployments. These include a current lack of unified Quality of Service (QoS) guarantees, making it difficult to maintain a consistent video experience across heterogeneous networks and devices. Moreover, frequent node online/offline behavior leads to unstable system topologies and unreliable transmission links. Additionally, as the number of participating nodes grows, achieving

efficient task coordination and link maintenance while preserving decentralization becomes increasingly complex. It is specifically to address these issues that this paper proposes a chain-structured transmission architecture. This design simplifies inter-node connection relationships through topological design and introduces lightweight fault detection and recovery mechanisms, thereby enhancing link stability and service continuity while ensuring system scalability. This offers a novel technical pathway for building efficient and reliable P2P multi-venue real-time broadcasting systems.

2.3. High-Quality P2P Video Streaming Systems

The central challenge for high-quality P2P video streaming systems lies in simultaneously meeting real-time, reliability, and security requirements within a decentralized environment. Researchers primarily explore three directions to achieve this: transport protocol optimization, network topology management, and Quality of Service assurance. The Celerity system serves as a prime example in this domain, designed for multi-party audio–video conferencing scenarios [36]. It utilizes an intelligent overlay construction algorithm, leveraging Round-Trip Time (RTT) and geographical distance to preferentially cluster nearby nodes. This is further coupled with machine learning analysis of historical link data to predict potential congestion and proactively adjust routing. By employing an anycast strategy, Celerity dynamically selects the path with the minimum end-to-end latency among multiple available options, thereby significantly enhancing video quality and user experience in large-scale conferencing environments.

Complementing such topological advancements is the strategic choice and optimization of the underlying transport protocol. The Secure Reliable Transport (SRT) protocol directly enhances UDP's low-overhead advantages by incorporating selective retransmission, effectively avoiding the head-of-line blocking issue prevalent in TCP [16,37]. SRT maintains stable throughput through dynamic congestion control, adaptive buffering, and intelligent traffic shaping. Furthermore, its design integrates end-to-end AES encryption, which, combined with enhanced ARQ and Forward Error Correction (FEC), robustly ensures data integrity and security. A key feature, the latency-tolerating window mechanism, allows the system to wait for packet retransmissions within a millisecond-level timeframe, enabling low-jitter real-time playback. In the context of a chain-based P2P topology, these intrinsic SRT characteristics not only alleviate the load on individual nodes but also maintain high-quality video transmission across multi-hop paths, thereby laying a robust technical foundation for scalable decentralized streaming systems.

As shown in Table 1, most existing P2P and hybrid systems exhibit higher end-to-end latency and slower recovery from failures, especially under dynamic network conditions or node churn. For example, SmoothCache and ALIVE, while effective in reducing server load, rely on mesh or hybrid topologies that introduce significant signaling and coordination overhead, resulting in recovery times exceeding 3 s and packet loss spikes above 1%. ReputeStream and Celerity improve resilience through reputation and overlay optimization but still suffer from multi-second recovery and non-negligible failure rates in large-scale or high-churn scenarios.

Table 1. Comparison of representative P2P and hybrid video streaming systems.

System	Topology	Latency (ms)	Recovery Time (s)	Failure Rate/Packet Loss
SmoothCache [31]	Mesh/Hybrid	800–1200	>5	∼2% (burst)
ALIVE [28,32]	Hybrid (P2P + Edge)	400–900	3–5	∼1%
ReputeStream [7]	Multi-layer P2P	300–700	3–4	>1% (under churn)
Celerity [36]	Mesh	250–600	2–4	∼1%
Proposed (Chain + SRT)	Chain	200–700	0.8–1.5	<0.2%

In contrast, the proposed chain-topology system, leveraging SRT and intelligent self-healing, achieves sub-second delays and maintains packet loss below 0.5% even in multi-hop deployments. The chain structure minimizes the number of control messages and simplifies failure localization, enabling rapid relay path reconstruction. SRT's selective retransmission and adaptive buffering further reduce latency and loss compared with TCP-based or mesh solutions. From a computational perspective, mesh and hybrid topologies (e.g., SmoothCache, ALIVE, Celerity) require each node to maintain multiple neighbor states, perform complex scheduling, and handle frequent signaling, leading to $O(n)$ or higher per-node overhead as the network scales. WebRTC-based systems, while offering NAT traversal and real-time media, incur additional CPU/memory costs for ICE negotiation, media encoding, and multi-peer synchronization. In contrast, the chain-based approach reduces per-node state to a single upstream and downstream connection, with failure detection and recovery logic that is both lightweight and deterministic. This design not only lowers computational and memory requirements but also enhances predictability and scalability for large deployments.

2.4. AI in Decentralized Video Streaming

The integration of artificial intelligence (AI) and machine learning (ML) techniques has emerged as a transformative approach for addressing the complex challenges in decentralized video streaming systems [38–40]. Recent advances in AI-driven video streaming focus on three primary areas: adaptive bitrate control, quality of experience (QoE) optimization, and intelligent resource management. Modern reinforcement learning (RL) approaches have demonstrated significant improvements in adaptive bitrate streaming. Chen et al. [41] proposed an edge-assisted RL framework that leverages distributed computing resources to optimize video delivery decisions in real-time, achieving up to 25% improvement in QoE metrics compared with traditional heuristic methods. Similarly, Liu et al. [42] developed a neural adaptive bitrate algorithm that uses deep Q-networks (DQN) to learn optimal bitrate selection policies from network conditions and user preferences, demonstrating superior performance in dynamic network environments.

The emergence of federated learning has opened new possibilities for collaborative optimization in distributed streaming systems. Wang et al. [43] introduced FedStream, a federated learning framework that enables edge nodes to collaboratively train adaptive streaming models while preserving privacy. This approach allows the system to benefit from collective intelligence across distributed nodes without centralizing sensitive data. Thompson et al. [44] further extended this concept by proposing federated adaptive bitrate streaming that leverages multi-agent reinforcement learning for coordinated decision-making across peer networks. Meta-learning techniques have also shown promise in addressing the heterogeneity and dynamics of decentralized streaming environments. Garcia et al. [45] developed a meta-reinforcement learning approach that enables rapid adaptation to new network conditions and user scenarios with minimal training data. This is particularly valuable in P2P systems where nodes frequently join and leave the network, requiring quick adaptation to changing topologies and conditions.

Quality of experience optimization has been enhanced through AI-driven approaches that consider multiple factors beyond traditional network metrics. Zhang et al. [46] proposed a QoE-aware RL algorithm for multi-path video streaming that jointly optimizes bitrate, buffer management, and path selection to maximize user satisfaction. Their approach incorporates perceptual quality models and user behavior patterns to make more informed streaming decisions. Edge intelligence has become increasingly important for real-time video processing and delivery. Ahmad et al. [47] introduced neural buffer management techniques that use recurrent neural networks to predict optimal buffer sizes

and prefetching strategies based on content characteristics and network dynamics. Patel et al. [48] developed intelligent edge caching mechanisms that use deep learning to predict content popularity and optimize cache placement for personalized video streaming. Within the context of our proposed chain-topology system, these AI techniques can be integrated to enhance autonomous operation and performance optimization. For instance, the fault detection mechanism described in our system could be augmented with neural anomaly detection models trained on historical network telemetry data. The adaptive relay path selection could benefit from RL algorithms that learn optimal routing decisions based on real-time network conditions and QoS requirements. Furthermore, federated learning approaches could enable collaborative optimization across the chain topology without requiring centralized coordination, maintaining the decentralized nature of the system while benefiting from collective intelligence [33].

2.5. Hardware Solutions for Decentralized Streaming

Recent advancements in low-power edge hardware have enabled the practical deployment of decentralized video streaming systems across diverse computational environments. The heterogeneous nature of modern edge computing infrastructure presents both opportunities and challenges for P2P streaming deployments. To evaluate the practical viability of our proposed system, we consider three representative hardware platforms that span the performance spectrum commonly found in real-world deployments. High-performance edge devices, exemplified by the AMD Ryzen 7 8845H processor adopted in this experiment, offer robust computational capabilities suitable for demanding streaming scenarios with multiple concurrent connections and high-resolution video processing [49]. The 8845H features eight cores/16 threads operating at a base frequency of 3.8 GHz (boost up to 5.1 GHz), integrated Radeon 780M graphics with hardware-accelerated H.264/H.265 encoding and decoding capabilities, and support for up to 4K@60fps video processing with dedicated media engines [50]. Its 45W TDP and advanced power management enable sustained high-performance operation while maintaining thermal efficiency, making it ideal for source nodes and high-traffic relay points. Mid-range platforms, such as the Intel N100, provide extraordinary performance-per-watt characteristics ideal for typical relay node deployments in resource-constrained environments [51]. The N100 operates at a 1.0 GHz base frequency (boost up to 3.4 GHz) with four cores/four threads, featuring Intel UHD Graphics with hardware acceleration for H.264/H.265 and AV1 codecs, capable of handling 1080p@60fps or even higher video streams with only 6 W TDP. Its efficient architecture delivers approximately 15–20 Mbps video relay throughput while consuming minimal power, making it suitable for battery-powered or solar-powered edge deployments. Lower-end devices, represented by the Intel Celeron J1900, which is usually used in IoT devices, embedded computing, or things like network gateways and industrial controllers, demonstrate the system's ability to operate on legacy or cost-optimized hardware [52]. Despite its modest specifications (4 cores at 2.0 GHz base, 2.4 GHz burst, 10 W TDP), the J1900 can effectively handle standard-definition to 720p video relay tasks through software-based codec processing, achieving 5–8 Mbps throughput with optimized SRT parameter tuning. This capability extends deployment accessibility to embedded devices like smart screens, IoT gateways, and legacy computing infrastructure, ensuring broad network participation.

The hardware-accelerated video processing capabilities across these platforms significantly reduce CPU utilization and power consumption compared with software-only implementations. Hardware decode acceleration reduces processing overhead by 60–80%, while dedicated media engines enable concurrent multi-stream processing [53]. This hardware diversity is crucial for practical P2P systems, where nodes with varying computational capabilities must collaborate effectively to maintain service quality. The proposed design

is compatible with heterogeneous hardware and performs effectively even on low-end devices, as will be demonstrated in our experimental evaluation.

3. System Design

3.1. System Design Overview

The decentralized multi-venue real-time video broadcasting system proposed in this paper adopts an innovative peer-to-peer architectural design. The core objective of the proposed system is to minimize system complexity and deployment costs while simultaneously ensuring low latency and high reliability. Fundamentally, the system is composed of three core components: the *Streaming Source*, the *Playback & Relay Node*, and a *Distributed IP Table Management* module.

The overall architecture is shown in Figure 2. To summarize the integration with the overall system architecture, our system's implementation involves the streaming source establishing an `srt://` listener via FFmpeg to receive video streams initiated by OBS or production software. The first node in the chain then operates in caller mode, actively connecting to the first downstream node. All subsequent relay nodes function in a listener–caller combination mode, sequentially relaying the stream. All nodes manage control signaling through WebSocket connections, handling node registration, link updates, and broken link repairs. When the chain topology changes, SRT connections are automatically negotiated, establishing a fully decentralized, stable, chain-based media relay system.

Figure 2. Overall system architecture design.

The system architecture adheres to a set of fundamental design principles that guide its construction and operation. Firstly, the *Principle of Simplicity* is achieved by employing a chain topology to reduce routing complexity, thereby avoiding the need for complex network discovery and maintenance mechanisms. Secondly, the *Principle of Reliability* is upheld through the integration of multi-layered fault detection and self-healing mechanisms, ensuring that single points of failure do not compromise overall service continuity. Thirdly, the *Principle of Scalability* is supported by enabling dynamic node joining and departure without requiring a complete network reconfiguration. Finally, the *Principle of Security* is paramount, with end-to-end encryption implemented across the entire link to guarantee the confidentiality and integrity of data transmission.

A crucial enabler for this system is its reliance on the *IPv6 Network Foundation*. The system fully leverages the technical advantages of the IPv6 protocol, where each participating

node acquires a globally unique IPv6 address via DHCPv6 or SLAAC mechanisms [23]. Compared with the intricate NAT traversal mechanisms prevalent in IPv4 environments, IPv6's native end-to-end connectivity significantly simplifies the P2P connection establishment process. Complementing this, the node discovery mechanism is based on a pre-configured list of seed nodes, allowing new nodes to register their IPv6 address and port information with a coordination server via HTTPS RESTful API.

This robust network foundation allows for the implementation of the *Chain Transmission Topology*. The system adopts a unidirectional chain structure, where the video stream originates from the streaming source, sequentially traverses intermediate relay nodes, and ultimately reaches the tail node of the chain. To formalize this, let the set of nodes be represented as $N = \{n_0, n_1, \ldots, n_k\}$, where n_0 is the streaming source and n_k is the chain tail node. The transmission path can thus be expressed as

$$Path = n_0 \xrightarrow{SRT} n_1 \xrightarrow{SRT} n_2 \xrightarrow{SRT} \cdots \xrightarrow{SRT} n_k \tag{1}$$

Each relay node n_i ($i \in [1, k-1]$) concurrently performs both reception and forwarding functions. The core algorithm governing a relay node's transmission behavior is detailed below (Algorithm 1):

Algorithm 1 Node Relay Transmission Algorithm

```
procedure RelayTransmission(upstream_addr, downstream_addr)
    listener ← CreateSRTListener(upstream_addr)
    caller ← CreateSRTCaller(downstream_addr)
    while connection_active do
        packet ← listener.receive()
        if packet ≠ null then
            caller.send(packet)
        end if
        CheckConnectionHealth()
    end while
end procedure
```

Based on the P2P chain and IPv6 foundation, the system employs a layered protocol stack design to organize its functionalities. At the application layer, responsibilities include video encoding/decoding and playback control. The SRT layer then provides crucial services such as reliable transport, congestion control, and secure data transfer. Beneath SRT, the UDP layer handles the underlying datagram transmission. Finally, the IPv6 layer offers fundamental network addressing and routing capabilities, forming the core communication backbone.

While the data transmission architecture is fundamentally decentralized, the system still requires a lightweight coordination mechanism to maintain the node list and topology information. The coordination server is designed with a RESTful API. It primarily performs node registration and deregistration via `POST/DELETE /api/nodes` endpoints, topology querying and updates using `GET/PUT /api/topology`, and health status monitoring accessible via `GET /api/health`. This setup ensures efficient management of the dynamic network without centralizing the actual data stream.

To further enhance system robustness and efficiency, a Load Balancing Mechanism is incorporated. This mechanism prevents any single node from bearing an excessive forwarding load by supporting parallel transmission across multiple chains. When a specific link is detected to be overloaded, the coordination server can dynamically create new transmission links to distribute the load. Load assessment metrics, which inform these

decisions, include CPU utilization, memory consumption, network bandwidth usage, and packet loss rate, enabling proactive and adaptive load management.

3.2. Transmission Mechanism

In real-time video streaming and broadcasting systems, the choice of transmission protocol directly dictates the system's latency characteristics, reliability assurance, and security level. The traditional TCP protocol, while providing reliable data transfer, introduces unpredictable latency variations due to its inherent congestion control mechanisms and ordered delivery requirements, especially under network jitter, making it difficult to meet the stringent demands of real-time transmission [54].

Our system strategically employs the Secure Reliable Transport (SRT) protocol as its core transport layer. SRT is built over UDP and achieves TCP-level reliability while preserving UDP's low-latency properties by implementing selective retransmission and intelligent congestion control at the application layer [16].

3.2.1. Design and Features of Integrated SRT Protocol

The SRT design includes optimized connection establishment and handshake procedures which enhanced stability and reliability of network connection. It utilizes a lightweight four-way handshake mechanism (INDUCTION-CONCLUSION). Compared with TCP's three-way handshake, SRT dedicates its first phase to negotiating crucial transmission parameters such as MTU, latency tolerance, and encryption configuration, with the second phase completing connection confirmation. This entire process is fully optimized for real-time transmission. Within our chain-based architecture, each node's listener–caller mode enables it to swiftly establish a new connection with a fallback node upon detecting an upstream link failure. This connection establishment typically falls within 50–100 ms, significantly faster than the multi-second reconnection times often seen in traditional P2P systems.

Furthermore, SRT incorporates a sophisticated bidirectional adaptive flow control mechanism. This intelligent stream control is driven by receiver-side feedback, continuously monitoring receive buffer occupancy, network RTT variations, and packet loss patterns to dynamically adjust the sender's data transmission rate. Specifically, the receiver periodically sends ACK packets within a defined time window, carrying information about its current buffer status and network quality assessment. The sender then uses this feedback to calculate an optimal sending window, formulated as

$$SendingRate_{optimal} = \min\left(\frac{BufferSpace}{RTT}, BandwidthEst \times (1 - LossRate)\right) \quad (2)$$

where $BufferSpace$ denotes the buffer space available at the receiver and $BandwidthEst$ represents the estimated bandwidth. This mechanism is particularly vital in multi-hop chain transmission, effectively preventing sudden traffic bursts from upstream nodes from overwhelming downstream node buffers.

SRT also introduces the concept of latency tolerance, allowing the system to wait for retransmission of lost data packets within a specified time window. Packets are discarded if this timeout is exceeded, ensuring real-time integrity. Our system adopts a hierarchical latency configuration strategy, adapting to link hop count and geographical distribution features:

$$Latency_{hop_i} = BaseLatency \times (1 + DistanceFactor_i + HopPenalty_i) \quad (3)$$

In this equation, $BaseLatency$ signifies a fundamental latency setting (typically 120–200 ms), $DistanceFactor_i$ is a geographical distance correction factor, and $HopPenalty_i$ accounts for

multi-hop cumulative penalty. This precise latency control enables the system to minimize end-to-end delay while ensuring smooth video playback.

3.2.2. Adaptive Data Transmission at the Protocol Layer

The protocol benefits from an enhanced ARQ retransmission strategy that combines selective retransmission with Forward Error Correction (FEC). For critical video frames, such as I-frames and important P-frames, the system automatically adds redundant encoding. For regular data packets, a rapid retransmission mechanism is employed. The retransmission decision is intelligently made based on network conditions and content importance:

$$RetransDecision = \begin{cases} FEC + ARQ & \text{if } FrameType = I \text{ and } LossRate > \theta_{high} \\ ARQ & \text{if } LossRate \leq \theta_{low} \\ Drop & \text{if } Age > LatencyThreshold \end{cases} \quad (4)$$

This strategy ensures the reliable transmission of critical video data while preventing excessive retransmissions from impacting real-time performance. SRT's adaptive retransmission mechanism specifically utilizes the Selective Repeat ARQ algorithm, which retransmits only lost packets, effectively bypassing the head-of-line blocking problem inherent in traditional TCP [37,55]. The size of the retransmission window dynamically adjusts according to the network RTT and the packet loss rate, following the formula [56,57]:

$$W_{retrans} = \min(W_{max}, \alpha \cdot RTT \cdot BDP + \beta \cdot Loss_{rate}) \quad (5)$$

where W_{max} denotes the maximum window size, BDP is the bandwidth-delay product, and RTT is the round-trip time, multiplied by BDP to quantify the amount of data that can be in transit in the network. α and β are tuning parameters that scale the contribution of bandwidth-delay product and loss, respectively. These are usually determined based on network conditions or system design goals. This mechanism ensures efficient retransmission under complex network conditions, making it particularly suitable for multiple transmission scenarios. Within the system's chain architecture, each node independently and dynamically maintains its retransmission window, preventing upstream packet loss from causing cascading impacts on downstream nodes. The values of tuning parameters α and β directly influence the responsiveness of the retransmission mechanism. A higher α emphasizes round-trip time (RTT), favoring stable networks where delay reflects congestion more reliably, while β governs sensitivity to packet loss to ensure smooth transmission under poor connectivity. In noisy or high-jitter environments, increasing α relative to β effectively extends the temporal smoothing of retransmission decisions—analogous to lengthening the signal period and reducing information density—to suppress transient noise and prevent overreaction. In practice, when using ffmpeg, the parameters are controlled indirectly via setting parameters, including maximum acceptable latency, overhead bandwidth, and input bandwidth. These parameters trigger the built-in fine-tuning of the SRT protocol's congestion control and retransmission mechanisms, allowing the system to automatically adaptively optimize performance based on real-time network conditions.

Dynamic congestion control is another key feature of SRT, implementing a hybrid algorithm based on latency and packet loss to maximize throughput while maintaining low latency. The sending window adjustment policy is as follows:

$$W_{new} = \begin{cases} W_{current} + 1 & \text{if } RTT < RTT_{target} \text{ and } Loss < Loss_{threshold} \\ W_{current} \times 0.875 & \text{if } RTT \geq RTT_{target} \text{ or } Loss \geq Loss_{threshold} \end{cases} \quad (6)$$

This algorithm dynamically adjusts the sending rate by monitoring in real-time the conditions of the network. Compared with traditional TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm, SRT's congestion control is more aggressive, allowing it to adapt quickly to network changes. In practical deployment, our system sets different values for RTT_{target} based on the link hierarchy and geographical distance, ensuring optimal transmission performance for each hop.

SRT protocol also implements a predictive adaptive buffer management mechanism. The receiver's buffer size dynamically adjusts based on network jitter and packet loss patterns:

$$BufferSize = BaseSize + \gamma \cdot \sqrt{Var(RTT)} + \delta \cdot E[Loss_{burst}] \quad (7)$$

where $Var(RTT)$ represents the variance of RTT, $E[Loss_{burst}]$ is the expected value of burst loss, and γ and δ are weighting coefficients. This dynamic buffering mechanism effectively balances latency and reliability requirements, ensuring smooth playback even when network conditions deteriorate.

3.2.3. Security Mechanisms and Reliability Assurance at the Protocol Layer

For security, SRT supports end-to-end security mechanisms via AES-128/256 encryption algorithms. Key exchange is handled using a preshared key (PSK) mode, which ensures the confidentiality and integrity of data transmission [58]. The cryptographic handshake employs a four-way handshake protocol. This involves the Caller sending an INDUCTION request with encryption capability negotiation information, followed by the Listener responding to the INDUCTION, confirming encryption parameters and session ID. Subsequently, the Caller sends a CONCLUSION request containing the PSK-derived session key, and finally, the Listener responds to the CONCLUSION, establishing the encrypted channel. To adapt to complex security environments involving multiple venues and public networks, our system independently negotiates encryption keys for each link. This design ensures that even if one node is compromised, the security of the entire link remains unaffected.

3.2.4. Protocol Performance Optimization and Quality Assurance

To further enhance the SRT protocol's performance in chain transmission, our system integrates several protocol performance optimization and quality assurance strategies. Firstly, a timestamp synchronization mechanism is employed to ensure temporal consistency across multi-hop transmissions. Secondly, adaptive parameter tuning is implemented based on network path characteristics, dynamically optimizing key parameters such as latency and retransmission timeouts according to link hierarchy, geographical distance, and current network conditions. Lastly, a traffic shaping mechanism, utilizing a token bucket algorithm, smooths out burst traffic, preventing network congestion from impacting downstream nodes.

In the context of P2P systems, quality management mechanisms are crucial, and node reputation management stands out as a significant approach. The ReputeStream system [7] introduces a reputation management mechanism based on a multi-layer architecture, which inspires the design of this system.

Reputation Calculation Model

In this system, each node maintains a reputation value that reflects its reliability and performance in the network. The reputation value is computed based on the node's historical interaction data, including successful and failed transmissions, as well as feedback from other nodes. This approach allows the system to dynamically adjust the topology based on

node reliability, ensuring that high-reputation nodes are prioritized for critical transmission tasks. This model employs Bayesian inference to compute node reputation values:

$$R_i(t+1) = \alpha \cdot R_i(t) + (1-\alpha) \cdot \frac{S_i(t)}{S_i(t) + F_i(t)} \tag{8}$$

where $R_i(t)$ is the reputation value of node i at time t, and $S_i(t)$ and $F_i(t)$ represent the counts of successful and failed interactions, respectively. This model effectively identifies and penalizes malicious nodes, thereby enhancing overall system reliability. In our system, reputation values are dynamically used to adjust a node's position within the chain topology, prioritizing high-reputation nodes for critical transmission tasks.

Layered Topology Optimization

The system constructs a layered overlay based on node reputation values, where high-reputation nodes are strategically positioned closer to the content source to form a stable transmission backbone network [29]. The objective function for the simple chain topology optimization is

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} w_{ij} \cdot d_{ij} \cdot (1 - R_j) \tag{9}$$

with the constraints

$$\begin{cases} \sum_{j=1}^{n} w_{ij} = 1 \\ w_{ij} \in \{0, 1\} \end{cases} \tag{10}$$

where w_{ij} denotes the traffic weight from node i to node j, whose potential values are just 0 or 1; d_{ij} is the distance reflected by factors such as ping latency, physical distance, bandwidth, and network accessibility; and R_j is the reputation value of node j. By solving this optimization problem, the system can construct an optimal transmission topology that considers both network performance and node reliability. This cost function balances path distance and node reliability. We also tested alternative forms using hop count or delay-only metrics, but the proposed function showed superior performance in maintaining stream quality under dynamic conditions.

Dynamic Load Balancing and Fault Prediction

Based on historical performance data and real-time monitoring metrics, the system implements a predictive load balancing mechanism using a hybrid Multi-layer Perceptron (MLP) with Long Short-Term Memory (LSTM) architecture. LSTMs are specialized recurrent neural networks designed to capture long-term dependencies in sequential data through a sophisticated gating mechanism that selectively remembers or forgets information [59]. This capability is crucial for analyzing temporal patterns in network performance metrics, where historical trends strongly influence future behavior. MLPs, in contrast, are traditional feedforward neural networks that excel at non-linear classification and regression tasks [60]. By introducing LSTM layers before the MLP classifier, our system gains the ability to detect subtle temporal anomalies in resource utilization patterns—such as gradually increasing memory consumption or cyclical network congestion—that simple threshold-based or non-recurrent models would miss. This hybrid architecture effectively addresses the temporal nature of network failures, which typically manifest as evolving patterns rather than instantaneous events, perfectly aligning with the system's need for proactive fault detection and recovery.

The system collects data every 50 ms, incorporating the previous failure probability and current resource utilization metrics to build a comprehensive dataset for model training and online adaptation:

$$P_{failure}(t) = \text{MLP}(\text{LSTM}([P_{failure}(t-1), CPU(t), Mem(t), Net(t)])) \tag{11}$$

where the LSTM component captures temporal dependencies in the sequence $[P_{failure}(t-1), CPU(t), Mem(t), Net(t)]$, and the MLP performs final classification. The model employs transfer learning based on the assumption that resource-failure relationships remain consistent across similar devices in short time periods, enabling effective knowledge transfer across the deployment fleet. Online learning continuously adjusts model parameters through comparative learning against pre-trained baselines, with the update rule:

$$\theta_{t+1} = \theta_t - \boldsymbol{\alpha} \nabla L(\text{ground truth}, P_{failure}(t)) - \boldsymbol{\beta} \nabla L_{transfer}(\theta_{pre}, \theta_t) \tag{12}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ control the online learning rate and transfer learning regularization, respectively. When the predicted probability exceeds a predefined threshold, the system proactively triggers link reconstruction to ensure service continuity.

Thus, at the protocol design level, our system fully leverages the extremely low transmission latency advantages of the UDP protocol, combined with the reliability, security, and intelligent flow control capabilities provided by the SRT protocol. This synergy achieves efficient and stable multi-hop chain-based real-time stream broadcasting in complex public network environments, effectively supporting dynamic chain topology reconstruction and self-healing from link breaks. This provides a robust transmission protocol foundation for large-scale, low-cost, real-time, multi-venue video systems.

3.3. Fault Detection and Link Self-Healing

Despite significant technical advancements in the protocol design level, there remains room for improvement in several areas. Specifically, fault recovery mechanisms in complex topologies require optimization, delay control for long-distance, cross-regional transmissions needs further enhancement, and service quality assurance mechanisms in heterogeneous device environments need strengthening. Our proposed chain-based architecture addresses these challenges by simplifying topological complexity and integrating the advantages of the SRT protocol, offering an effective technical solution.

While the chain transmission topology offers simplicity and efficiency, its sequential nature means that the failure of any single node can potentially interrupt service for all downstream nodes. To address this critical challenge, this paper designs a robust fault recovery mechanism based on multi-layered detection and intelligent reconstruction [7].

The system employs a three-layered fault detection architecture to ensure rapid and accurate fault localization. Firstly, application-layer heartbeat detection is implemented, where each node sends periodic heartbeat signals to the coordination server via a WebSocket long connection. The heartbeat interval adapts dynamically:

$$HeartbeatInterval = BaseInterval \times (1 + \alpha \times NetworkJitter) \tag{13}$$

where $BaseInterval$ is the base heartbeat interval, α is an adjustment factor, and $NetworkJitter$ represents the network jitter metric.

Secondly, transport-layer connection monitoring is continuously performed by the SRT protocol layer, which assesses link health through various metrics:

$$HealthScore = w_1 \times (1 - LossRate) + w_2 \times \frac{1}{RTT_{norm}} + w_3 \times \frac{1}{Jitter_{norm}} \tag{14}$$

where w_1, w_2, w_3 are weighting coefficients, and RTT_{norm} and $Jitter_{norm}$ are normalized RTT and jitter values.

Finally, data stream integrity detection within the SRT protocol combines Selective Repeat ARQ with a latency-threshold-based discarding mechanism. Its core process involves each receiver maintaining a receive window, where incoming packets are arranged by sequence number. If a sequence number gap is detected (i.e., $Seq_{i+1} > Seq_i + 1$), it is identified as a packet loss, and a NAK (Negative ACKnowledgement) request is immediately sent to the sender for retransmission of the missing packet. Upon receiving a NAK, the sender only retransmits the specified sequence number, avoiding full-window retransmissions and enhancing efficiency. Concurrently, to ensure real-time performance, SRT assigns a send timestamp T_{send} to each data packet. The receiver calculates the difference between the current time T_{now} and T_{send}. If the waiting time T_{wait} for a lost packet to be retransmitted exceeds the system's maximum allowed latency threshold L_{max}, the system proactively discards that packet, no longer awaiting retransmission, to prevent stale data from affecting subsequent smooth playback.

The algorithm pseudo-code for this mechanism is as follows (Algorithm 2):

Algorithm 2 SRT Packet Loss Detection and Proactive Discarding Algorithm

```
for each packet pkt in the receive window do
    if pkt is lost then
        Send NAK(pkt.seq) to request retransmission
        T_wait ← T_now − pkt.T_send
        if T_wait > L_max then
            Proactively discard pkt, no longer waiting for retransmission
        end if
    end if
end for
```

The key calculation for packet waiting time is $T_{wait} = T_{now} - T_{send}$, and the criterion for proactive discarding is

$$\text{If } T_{wait} > L_{max}, \text{ then discard the packet} \tag{15}$$

where L_{max} is the maximum latency tolerance threshold configured in the system (e.g., 120 ms, 200 ms). Through these mechanisms, SRT maximizes packet recovery while ensuring real-time performance, preventing outdated packets from impacting overall smoothness, and effectively enhancing video quality and stability in chain multi-hop transmission.

Upon detection of node failure, the system initiates a *dynamic topology reconstruction algorithm*. This process first involves *fault impact assessment*, evaluating the scope of the affected nodes:

$$ImpactScope = \{n_i | i > FailedNodeIndex \text{ and } n_i \in ActiveNodes\} \tag{16}$$

Following this, the optimal reconstruction path calculation determines the best new path based on the network status and load of the remaining nodes:

$$OptimalPath = \arg\min_{path} \sum_{i=1}^{|path|-1} w_{delay} \times RTT_{i,i+1} + w_{load} \times Load_i \tag{17}$$

where w_{delay} and w_{load} represent the weights for delay and load, respectively.

To ensure continuous service, the system implements a multi-level fallback self-healing strategy. This includes local buffer playback, where each node maintains a circular buffer to provide short-term cached playback if the upstream connection is interrupted:

$$BufferDuration = \max(MinBuffer, \frac{AvailableMemory}{VideoDataRate} \times SafetyFactor) \quad (18)$$

In addition, the system uses backup link switching. Pre-established backup transmission links are quickly activated if the primary link fails. The algorithm for backup link switching is described as follows (Algorithm 3):

Algorithm 3 Backup Link Switching Algorithm

```
procedure BACKUPLINKSWITCHING
    if primaryLinkFailed then
        backupLink ← selectBestBackup()
        establishConnection(backupLink)
        seamlessSwitch(primaryLink, backupLink)
        notifyDownstream(newUpstreamInfo)
    end if
end procedure
```

The system's fault recovery performance metrics were obtained from output logs and task manager monitoring during experiments. In a specific test case as an example, fault detection delay was measured as $T_{detection}$ (time from breakdown to interrupt trigger), topology reconstruction delay as $T_{reconstruction}$ (time for reconnection and chain rebuilding), and service recovery delay as $T_{recovery}$ (time until video playback resumed normally). The total fault recovery time is therefore calculated as

$$T_{total} = T_{detection} + T_{reconstruction} + T_{recovery} \quad (19)$$

Through the synergistic operation of these mechanisms, the system maintains high availability and service continuity in complex network environments, effectively addressing various fault scenarios.

As shown in Figure 3, the fallback self-healing mechanism is triggered when a link break occurs, which is detected by the SRT protocol layer through packet loss and timeout monitoring. Each node will immediately signal the coordination server to report that it is still online and ready for a new connection right after the link break is detected. The video streaming is down soon after all buffer data is consumed, but the video player process will keep waiting until a new video stream is received. No crash will happen during this period, and the video player will not be interrupted. The new chain is reconstructed by the coordination server and broadcast to all nodes, which will then establish new SRT connections. The video streaming process of each node will automatically switch to the new chain and stream the received video data to the new downstream nodes. The video player will then resume playback. This procedure ensures that the system can quickly recover from link breaks and continue to provide uninterrupted video streaming services.

The entire response process of this fallback mechanism is designed with latency optimization in mind. Fault confirmation is typically completed within 10 to 100 ms of detecting a WebSocket link break. Subsequently, chain reconstruction and the dissemination of jump-point control commands consume approximately 100 to 300 ms. This, combined with the SRT layer's rapid caller-listener rehandshake process taking about 500 ms, ultimately allows the system to achieve link self-healing and resume playback within a 1 to 2 s window in most anomaly cases, which is significantly faster than traditional methods, which usually take minutes and require manual intervention. Through this design, the

system accomplishes multiple fault tolerance objectives: rapid isolation of local faults, swift link break convergence, minimal reconstruction windows, and uninterrupted business playback switching. Even in extreme network conditions involving multi-hop consecutive failures, the system can maintain stable operation of remaining links by relying on real-time feedback, thereby significantly enhancing overall system availability and business continuity assurance.

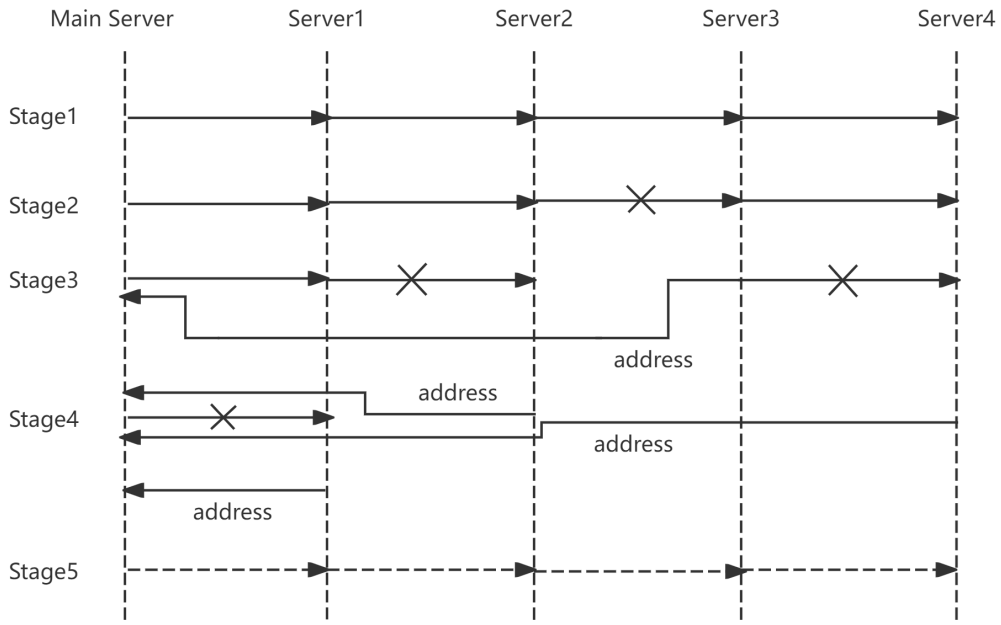


Figure 3. Chain link break recovery and fallback self-healing mechanism diagram.

3.4. Security Protection Mechanisms

Security is a foundational consideration in the design of our system. Unlike traditional centralized streaming platforms, which aggregate all audio and video traffic through a small number of servers—creating single points of failure and attractive targets for attacks—our system employs a decentralized peer-to-peer (P2P) chain architecture. In this design, video streams are forwarded sequentially through direct, one-to-one connections between client nodes. The central server is only responsible for node registration, link scheduling, and anomaly monitoring; it does not participate in the forwarding of any audio or video data. This approach eliminates the risks associated with data aggregation, centralized server leaks, or targeted attacks, thereby decentralizing transmission channels and significantly enhancing overall system security and resilience.

Each node in the chain listens on a local port instead of a remote one and only accepts connections initiated by its designated upstream node. The connection is strictly one-to-one, using the receiver's IPv6 address and port. There is no publicly exposed address that can be accessed arbitrarily. All chain connections are established in a fully controlled and authenticated manner, preventing any unauthorized access. As a result, the system is inherently secure by design. Eavesdropping or tampering with the data stream is virtually impossible, as all nodes would only establish connections with their designated upstream nodes, and the data stream has no chance of being accessed by any third-party nodes or servers.

At the data link layer, the system fully integrates the SRT (Secure Reliable Transport) protocol as its streaming media transport standard. The SRT protocol inherently embeds advanced symmetric encryption algorithms, including AES-128 and AES-256. Users can negotiate keys during link establishment, ensuring that every segment of video data is encrypted before being transmitted over the public network. In its specific implementation, all pushing and pulling nodes are serialized through the `srt://` protocol. The SRT protocol stack automatically encrypts and decrypts data packets during transmission and reception at each hop. Even if a malicious third party or "Man-in-the-Middle" (MITM) attempts to eavesdrop on or intercept data within the link, all transmitted content appears as high-strength ciphertext due to the lack of the correct session key, thereby guaranteeing the confidentiality and integrity of the content. Furthermore, because the system employs chain-based peer-to-peer forwarding, the actual data stream is not obtained directly from a server by the client. Instead, the server delivers the initial stream to the head node of the chain, which then pushes it to downstream nodes. Each level only needs to establish an SRT-encrypted connection with its direct upstream and downstream counterparts. The client's local FFmpeg decoding module only accesses and decrypts locally received stream data, eliminating the need for additional plaintext data interaction with the server. This chain-based push-pull separation significantly reduces the possibility of attackers injecting, forging, or hijacking content by impersonating servers or tampering with relay links, further enhancing the overall security protection capabilities of the system.

To specifically counter Man-in-the-Middle (MITM) attacks, the SRT protocol not only relies on key exchange and data encryption but can also incorporate identity verification using passwords or keys configured locally on the client. Only authenticated nodes are permitted to participate in the chain forwarding. Should an attacker attempt to forge a link node for hijacking, they cannot access the system's actual forwarding chain without obtaining the session key or successfully completing the server's legitimate registration process. Concurrently, during link breaks, automatic reconstruction, or fallback forwarding, all newly established sessions automatically inherit the original encryption parameters, ensuring that data security is not compromised throughout the re-connection process. Within the entire system, the server-side signaling communication utilizes WebSocket long connections. While TLS is not enabled by default for these connections, all critical control messages do not contain unencrypted audio or video data and can be further upgraded to encrypted transmission (e.g., using `wss://`) based on specific deployment environment requirements. The physical decoupling of the control plane and data plane means that even if the signaling plane were to be compromised, an attacker would be unable to directly access or manipulate the actual video stream content.

3.5. System Scalability and Network Topology Extensions

The chain-based architecture of the proposed system offers exceptional scalability characteristics that differentiate it from traditional centralized streaming solutions. Theoretically, the chain topology can be extended indefinitely, with each additional node contributing minimal overhead to the overall system while expanding its reach. This linear scaling property stands in stark contrast to centralized solutions, where increasing viewer capacity often incurs exponential cost increases in server infrastructure, bandwidth provisioning, and operational complexity.

A particularly powerful aspect of the system's design is its inherent support for topology transformation and branching. Any node within an existing chain can serve as an origination point for a new chain by using its locally received stream as input, effectively

creating a tree-structured network from the fundamental chain building blocks. This capability enables several significant advantages:

$$T(n) = \begin{cases} O(1) & \text{for adding a new node to existing chain} \\ O(log(n)) & \text{for establishing optimal tree depth with } n \text{ nodes} \end{cases} \quad (20)$$

First, tree structures provide more efficient distribution paths compared with single long chains, reducing cumulative latency and enhancing viewership scalability. Second, this arrangement facilitates localized viewing clusters within LANs or geographically proximate networks, where a single entry point into a regional network can serve multiple downstream viewers through local branching. Third, network administrators can deliberately design the topology based on specific performance objectives, geographical constraints, or audience distribution patterns.

Furthermore, this approach enables targeted optimization strategies. For instance, high-capacity nodes can be positioned at critical branch points to serve multiple downstream chains, while resource-constrained environments can be accommodated through careful branch placement and stream parameter adjustment. The system's flexibility allows implementers to minimize chain length where latency is critical while maximizing distribution breadth where coverage is the priority.

This inherent extensibility presents a fundamental economic advantage: while traditional centralized solutions encounter bandwidth bottlenecks and processing limitations that require costly hardware upgrades or cloud service tier increases, our decentralized approach requires only the addition of standard nodes—which can be implemented on commodity hardware—to expand capacity. This results in near-linear cost scaling rather than the super-linear or exponential cost growth typical of centralized architectures under increasing load.

Experimental validation confirms this scalability model, with successful deployments maintaining consistent performance characteristics across varied topologies ranging from simple 8-hop chains to complex tree structures with multiple branch points serving over 20 concurrent viewers. The architecture's ability to dynamically reorganize these topologies further enhances its adaptability to changing network conditions and audience requirements.

3.6. Stream Processing Flexibility and Format Adaptation

A significant advantage of the decentralized node architecture is that each participant in the chain not only receives the video stream but can also perform local processing operations according to specific requirements. This capability enables a wide range of applications beyond simple viewing, substantially enhancing the system's flexibility and utility across diverse use cases.

At each node in the transmission chain, the locally received video can be concurrently processed for multiple purposes. Local recording functionality allows participants to create persistent archives of the stream for later review or distribution, effectively enabling time-shifted viewing without centralized storage infrastructure. This is particularly valuable in educational contexts where lecture recordings can be made available asynchronously or in conference scenarios where presentations can be archived for reference. The implementation utilizes FFmpeg's container multiplexing capabilities to generate standard-format recordings (Algorithm 4):

Algorithm 4 Local Stream Recording and Processing

```
procedure LocalProcessing(incoming_stream)
    viewer ← CreateLocalViewer(incoming_stream)
    recorder ← CreateRecorder(incoming_stream, format)
    if downstream_requirements ≠ null then
        transcoder ← ConfigureTranscoder(incoming_stream, downstream_requirements)
        adapted_stream ← transcoder.process()
        ForwardStream(adapted_stream)
    else
        ForwardStream(incoming_stream)
    end if
end procedure
```

Furthermore, the system supports intelligent transcoding capabilities that can adapt the stream characteristics based on downstream requirements. This adaptive behavior is particularly valuable in heterogeneous environments where downstream nodes may have varying processing capabilities, connectivity constraints, or compatibility requirements. For example, when a high-performance node serves as an upstream source for resource-constrained devices (such as legacy hardware or mobile devices), it can dynamically transcode the stream to reduce computational demands on receivers:

$$TranscodingProfile = \begin{cases} \{codec : \text{AV1/HEVC}, bitrate : \text{high}\} & \text{for high-performance receivers} \\ \{codec : \text{H.264}, bitrate : \text{medium}\} & \text{for mid-range devices} \\ \{codec : \text{H.264}, bitrate : \text{low}, res : \text{720p}\} & \text{for low-power devices} \\ \{format : \text{FLV/RTMP}\} & \text{for legacy compatibility} \end{cases} \tag{21}$$

This capability enables complex adaptive topologies where content characteristics evolve as they traverse the network. For instance, a high-bitrate AV1-encoded 4K source stream might be maintained through the high-capacity backbone of the network, while branches serving different audience segments automatically transcode to more appropriate formats such as H.264 with lower bit rates for general viewers, reduced resolution for mobile devices, or even legacy formats like FLV for environments with older playback infrastructure.

The transcoding operation itself can be parameterized according to specific quality and efficiency requirements. For example, the encoding preset can be tuned based on the node's available computational resources:

$$EncodingPreset = \begin{cases} \text{veryslow} & \text{if } CPU_{available} > 80\% \text{ and not latency-sensitive} \\ \text{medium} & \text{if } 50\% < CPU_{available} < 80\% \\ \text{ultrafast} & \text{if } CPU_{available} < 50\% \text{ or latency-critical} \end{cases} \tag{22}$$

As a comparison, in traditional video streaming solutions, including RTMP, RTSP, or M3U8-MPEGTS, each receiver has to rely on the server to perform all transcoding operations. For instance, modern hardware for video processing, including integrated GPUs in Intel® N100 or AMD® Ryzen™8845H used in this experiment [49,51], as well as newer dedicated GPUs, supports efficient AV1 hardware decoding, allowing video playback with minimal CPU overhead, which is sometimes even negligible. In contrast, older Intel processors lack hardware decoding for modern codecs, forcing them to rely on software decoding, which can consume a significant portion of their limited performance, frequently exceeding 60% utilization and sometimes maxing out the CPU. Additionally, different GPUs vary in their decoding capabilities; while some can barely handle 1080p 60fps

H.264 at their limit, others effortlessly decode 4K or higher resolutions with more complex codecs like HEVC, VP9, or AV1. To ensure compatibility across devices, video sources are typically encoded in multiple bitrates and resolutions (e.g., 240p, 480p, 720p, 1080p, 2160p, or even higher, with varying color depth, frame rates, and certainly different bitrates). However, maintaining these parallel streams places an enormous burden on servers, as most consumer-grade GPUs can only process one or two simultaneous transcodes before reaching their limit, leading to unsustainable server loads. Additionally, traditional peer-to-peer (P2P) streaming solutions, even when integrated into CDN-like architectures, such as WebRTC or QUIC used for video sharing, face inherent limitations such as firewall restrictions, NAT traversal issues, ISP-imposed upload bandwidth caps, complex and dynamic network topologies, and constantly changing upload and download conditions. These challenges make P2P-based delivery unreliable and unpredictable, ultimately forcing streaming platforms to remain heavily dependent on centralized server infrastructure despite its scalability constraints and cost implications.

In contrast, our system's chain-based architecture allows each node to independently handle transcoding and processing tasks, effectively distributing the computational load across the network. This design not only alleviates the burden on any single node but also enables real-time adaptation to varying device capabilities and network conditions. This flexibility allows the system to balance quality and performance dynamically; for example, for devices with powerful video processing capabilities, the adopted codecs can be more advanced (e.g., AV1 or HEVC) to achieve higher compression efficiency and ensure smooth playback, keeping better video quality under a given bandwidth. Conversely, for devices with limited processing power, the system can automatically switch to more compatible formats like H.264 or even legacy codecs by sacrificing the resolution or bitrate, ensuring that all viewers can access the content without overwhelming their hardware.

This intelligent adaptation extends beyond mere format conversion to encompass advanced processing such as the following:

- Dynamic resolution scaling based on network conditions.
- Audio and video normalization and enhancement for improved intelligibility.
- Automated caption generation or overlay for accessibility.
- Video composition for multi-source presentations.
- Custom graphics insertion for branding or informational purposes.

In practical deployments, this flexibility has enabled novel applications such as multi-venue educational broadcasts where a single high-quality source stream is adapted to serve both modern smart classrooms (receiving full-quality HEVC) and legacy computer labs (receiving compatible H.264), all while maintaining local recordings at key administrative nodes. Similarly, in conference settings, presentation streams can be simultaneously recorded at full quality for archives while being transcoded to bandwidth-efficient formats for remote participants on varying connection types.

The decentralized nature of this processing architecture distributes the computational load across the network, avoiding the bottlenecks that would occur if all transcoding were performed at a central server. This approach aligns with the overall system philosophy of resilient, scalable, and efficient resource utilization through intelligent distribution of both network and processing tasks.

3.7. Deployment Details and Implementation Environment

In the practical engineering implementation of this system, comprehensive consideration has been given to cross-platform compatibility, module decoupling, and security controllability. The core components are developed using Python 3.12, leveraging Python's efficient network I/O and process scheduling capabilities to implement high-level logic

such as signaling services, link control, node registration, and client management. The system's control plane features an asynchronous HTTP/WebSocket server, designed to efficiently handle multiple concurrent signaling communications, support long-lived connection maintenance, and facilitate real-time message pushing. The primary role of the signaling protocol encompasses registering venue nodes, negotiating and dynamically adjusting link topologies, performing fault detection, and issuing anomaly notifications.

For data plane processing, all media streams within the system, including pushing and forwarding, utilize *FFmpeg 7.1.1* as the underlying engine. FFmpeg, a mainstream open-source streaming media processing tool in the industry, supports a rich array of protocols such as SRT, UDP, RTMP, and MPEG-TS. It possesses robust capabilities for concurrent multi-stream pushing, data transcoding, real-time recording, audio–video synchronization, and various encoding/decoding operations. The Python component dynamically injects control commands and input/output parameters by invoking local FFmpeg executables as subprocesses. This design enables flexible orchestration of business processes like multi-stream pushing, forwarding, and playback. Throughout the media pipeline, FFmpeg, in conjunction with the system's chain P2P architecture, achieves full process control for node-level listener–caller SRT connections, stream forwarding, broken link reconnection, and local real-time recording.

Upon receiving the video stream locally, the system exposes the stream via a localhost service (such as a local UDP, MPEG-TS over M3U8, or another SRT endpoint). This enables flexible downstream processing, including but not limited to real-time monitoring, playback through video players (e.g., MPV, VLC), local recording, and on-the-fly transcoding. For example, the received stream can be forwarded to a local FFmpeg process for recording or transcoded to different formats and bitrates to accommodate heterogeneous device capabilities. Downstream nodes may also receive these transcoded outputs, allowing for adaptive streaming tailored to varying network conditions and hardware performance. This design enhances the flexibility and robustness of the video streaming pipeline, supporting real-time preview, time-shifted playback, and multi-format distribution, while also addressing the complexities of data link diversity and device heterogeneity.

To further enhance system robustness and ensure uninterrupted service, an emergency fallback mechanism is integrated into the architecture. In the event of an SRT data transmission failure or process crash, downstream nodes are designed to automatically initiate a fallback procedure. Upon detecting a primary link disruption, the affected node actively attempts to retrieve the video stream from the upstream node's locally exposed service interface (such as a UDP endpoint or HTTP port). The retrieved stream is then re-encoded if necessary and forwarded downstream, effectively reconstructing the broken link. This fallback strategy enables rapid recovery, typically within seconds, minimizing the impact of transmission failures. Rather than causing a complete interruption or client-side crash, the system may only experience a brief period of video or audio stutter. By leveraging local stream exposure and automated re-encoding, the fallback mechanism ensures seamless self-healing of the transmission chain and maintains business continuity. This design significantly improves overall system resilience, allowing for graceful degradation and quick restoration of service in the face of network or process anomalies.

For auxiliary communication between the HTTP service and nodes, the system employs the HTTP protocol to carry certain management functions, including IP list maintenance, task scheduling, and link status queries. Concurrently, WebSocket long connections are utilized for real-time dissemination of link topology changes and abnormal switching signals, ensuring physical isolation and logical synchronization between the data and control planes. Regarding node management and secure deployment, the system assigns an independent listening port and a local buffer directory to each client. All critical pro-

cesses, such as registration, chain reconstruction, fault recovery, and fallback switching, are meticulously logged to facilitate problem tracing and engineering debugging. For deployment, it is recommended that each venue node operate on Windows 10/11 or a mainstream Linux distribution environment, with Python 3.12 and FFmpeg 7.1.1 installed. Network security groups should be configured to permit necessary UDP/SRT ports. In actual operation, the main venue server is typically deployed on a publicly accessible node, while sub-venue clients can be deployed on various local or public network nodes. Both IPv4 and IPv6 network environments are supported, ensuring compatibility with a wide range of practical application scenarios.

To further enhance security and operational efficiency, the system supports multiple levels of permission assignment and key configuration. All SRT sessions can enable AES-128/AES-256 encryption, allowing users to set session keys according to their security requirements. Log directories, cache files, and recording outputs can be configured with persistent paths to meet data backup and compliance needs. Looking ahead, the system is also designed to integrate with cloud platform automation deployment solutions. This will facilitate the rapid engineering rollout of large-scale, multi-venue chain broadcasting in diverse application scenarios such as education, conferences, and sporting events.

4. Experimental Methodology

A comprehensive and rigorous experimental framework was designed to systematically evaluate the performance, scalability, and fault tolerance of the proposed peer-to-peer (P2P) multi-site video streaming system, which integrates chain topology and intelligent self-healing capabilities. To ensure high reliability and reproducibility, the experimental procedure was structured to meticulously analyze the impact of relay-chain length, network conditions, and various fault scenarios using standardized evaluation metrics and robust statistical analysis.

To provide a comparative baseline, this study replicates the system described in the paper by Yusuf et al. [61]. Building on this baseline, we evaluate video quality across the relay chains using Video Multi-Resolution Fidelity (VMRF), Peak Signal-to-Noise Ratio (PSNR), and VMAF at each relay node. These tests were performed under three representative network conditions: high bandwidth with low loss (20 Mbps, 0.1%), medium bandwidth with moderate loss (10 Mbps, 1%), and low bandwidth with high loss (2 Mbps, 5%).

4.1. Relay Chain Length Analysis

Experiments were conducted meticulously with different lengths of the relay chain, specifically evaluating configurations consisting of 2-hop, 5-hop, 8-hop, and 20-hop relay nodes. Each node chain configuration started from a high-performance source node, a laptop equipped with AMD Ryzen™ 7 8845H, progressing downstream through nodes with heterogeneous computing capacities, including multiple low-power Intel® N100 mini PCs and Intel® Celeron® J1900 embedded computing terminal devices, and desktop computers with 12th generation Intel® Core™ i3 hardware, 5th generation Intel® Core™ i5 processors with NVIDIA® Quadro™600, 8th generation Intel® Core™ i7 processors, legacy Intel® Xeon™ E3 series workstation processors, or AMD™Pro A-Series processors (often referred to as APUs). This strategic choice was made to reflect the inherent heterogeneity found in real-world deployments, effectively simulating realistic relay-chain conditions and allowing precise measurements of cumulative end-to-end (E2E) latency variations associated with chain length increments. The devices are deployed across multiple cities in different provinces, spanning regions such as East China (Shandong) and Northwest China (Ningxia). Their placement varies from being just meters apart in the same lab to hundreds of kilometers apart on separate campuses connected via dedicated fiber-optic LAN, or even

thousands of kilometers apart on the Internet under different ISPs, simulating real-world network conditions and geographical distribution.

A further objective of the experimental design was to evaluate the cost-effectiveness and hardware adaptability of the proposed decentralized chain-based architecture. Specifically, experiments were planned to assess system performance on low-specification hardware, including typical household or personal laptops or desktops, or even lower-performance devices that cost less than USD 150 each on average. The goal was to determine whether reliable real-time video broadcasting could be achieved without reliance on high-performance servers. This aspect of the experimental methodology was intended to validate the practical feasibility of large-scale deployments under constrained hardware budgets and operational costs and to compare the resource utilization efficiency of the decentralized approach against traditional centralized solutions.

4.2. Network Condition Simulation and Analysis

To robustly assess system adaptability to varying network environments, three distinct and carefully controlled network scenarios were implemented:

- High-bandwidth, low packet-loss environment (20 Mbps bandwidth, 0.1% packet loss).
- Moderate-bandwidth, moderate-loss environment (10 Mbps bandwidth, 1% packet loss).
- Low-bandwidth, high-loss environment (2 Mbps bandwidth, 5% packet loss).

These scenarios were methodically designed to represent common real-world conditions ranging from stable corporate LAN environments to challenging WAN scenarios with high network congestion. The control over network parameters was achieved using openWRT routers, which allowed precise emulation of bandwidth and packet loss characteristics. The latencies are captured using the method described in Figure 4. Video quality at each relay node was systematically captured using the video player, with Video Multi-Resolution Fidelity (VMAF) scores computed real-time to objectively quantify visual fidelity degradation under each network condition. The data collection procedure adhered strictly to standardized practices to guarantee accuracy and reliability. Each scenario was executed multiple times to ensure statistical significance and to account for potential variability in network performance, and the recorded VMAF scores were aggregated for comprehensive analysis.

4.3. SRT Packet Loss Handling Mechanism Evaluation

To validate the effectiveness of SRT's enhanced packet loss handling mechanisms, a controlled comparative experiment was designed to quantify the performance differences between three packet loss handling strategies:

- Default ARQ (TCP-like): Traditional Automatic Repeat reQuest with strict in-order delivery, where any lost packet causes subsequent packets to be buffered until the missing packet is retransmitted and received.
- SRT Selective ARQ: SRT's selective retransmission mechanism that allows out-of-order delivery of non-critical packets while prioritizing retransmission of essential frame data (I-frames, motion vectors). This is mostly performed when the packet loss rate is low and the network conditions are mostly stable but still suffer from occasional packet loss.
- SRT Adaptive Drop: SRT's intelligent packet dropping strategy that discards packets when their retransmission would violate latency constraints, combined with Forward Error Correction (FEC) for graceful degradation. This is mostly performed when the packet loss rate is high but the required latency is low, such as in real-time video conferencing or live streaming scenarios.

The experimental setup utilized a controlled network emulation environment using openWRT routers to simulate packet loss and latency variations. The SRT protocol was configured with given parameters to ensure consistent handling across all three strategies, and the performance of each strategy was evaluated under varying packet loss rates (0.5%, 1%, 2%, 5%, and 10%) to quantify their impact on end-to-end latency, playback continuity, and user experience.

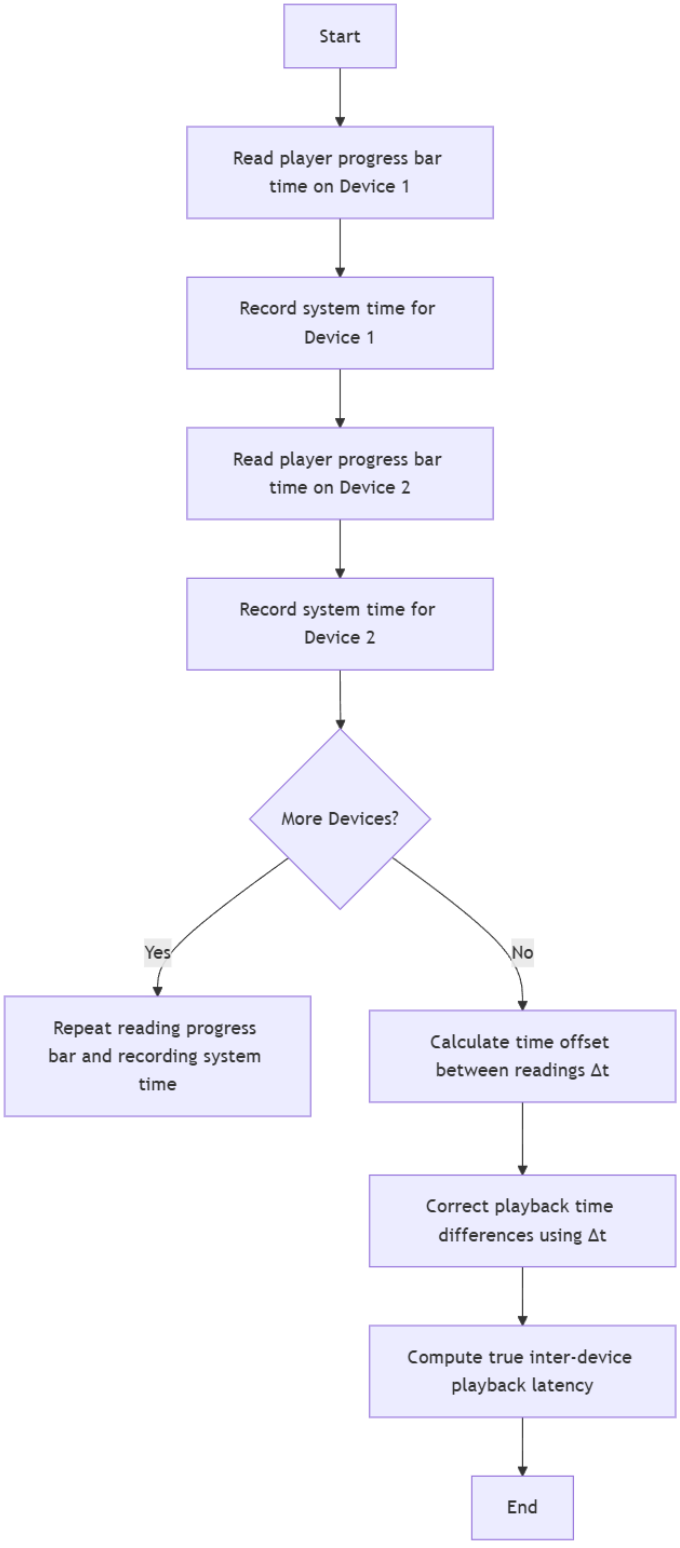


Figure 4. Delayed calculation flowchart.

4.4. Fault Resilience and Self-Healing Validation

Fault resilience was evaluated through carefully orchestrated simulations of realistic fault conditions, including instantaneous power disruptions lasting 30 s to assess rapid recovery capabilities, severe packet loss scenarios emulating significant network congestion events, and controlled process failures and software crashes to evaluate the robustness of the system's self-healing mechanisms. Recovery metrics such as mean recovery time and packet retransmission rates were rigorously documented. The effectiveness of the intelligent self-healing algorithm, featuring automated relay-chain reconstruction triggered by fault detection mechanisms, was validated through repeated testing across these scenarios. Statistical analysis was employed to ensure the significance and reliability of recovery performance outcomes.

Each experimental condition was replicated multiple times to ensure statistical validity and reduce variance in measurements. Data were aggregated, and statistical significance was quantified using confidence intervals and standardized statistical tests, ensuring robust and reliable conclusions aligned with the rigorous standards typical of IEEE and MDPI Applied Sciences publications.

5. Results

Under the medium network condition, the 5-hop chain achieved an average VMAF score of 83.5, significantly outperforming the baseline system's 65.2. Even under the most adverse network setting (2 Mbps with 5% packet loss), the proposed system maintained VMAF scores above 72 (Figure 5). The cumulative PSNR drop across 8 hops was only 2.1 dB, decreasing from 42.3 dB at the source to 40.2 dB at the terminal node, indicating well-preserved visual fidelity across the chain.

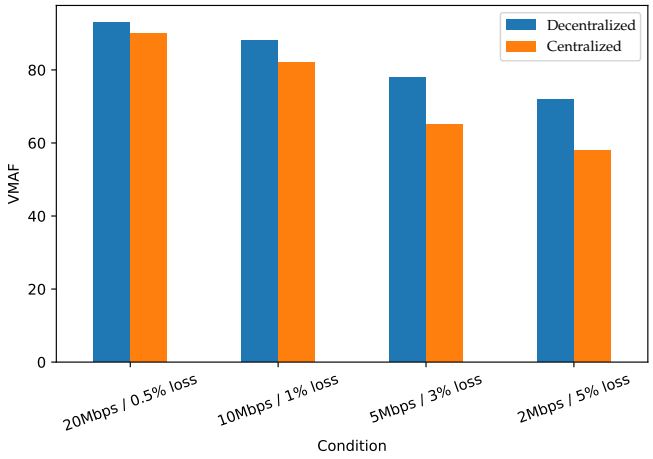


Figure 5. Comparison of VMAF: decentralized SRT chain vs. traditional centralized RTMP protocol.

The package loss handling performance of SRT was evaluated under various network conditions, and the results are shown in Figure 6. As is shown, the SRT protocol's adaptive strategies significantly outperformed traditional ARQ mechanisms, particularly under high packet loss rates.

Fault resilience was validated through a series of controlled fault injection tests. The system demonstrated robust self-healing capabilities, with an average recovery time of 12 s for a simulated 30 s power outage in some venues in the chain, a high loss spike broke the streaming continuity, or in a process crash, in a large-scale scenario. The standard deviation of recovery times across a single case was 1.2 s, which indicates that the recovery time is consistent and predictable across different fault scenarios and can be effectively managed by the system. On smaller scales, the values are much lower and the distinction is much

less pronounced, thus not discussed. In small-scale experiments involving only three hops, the recovery time was 1.3 s with a standard deviation of 0.2 s, which is negligible.

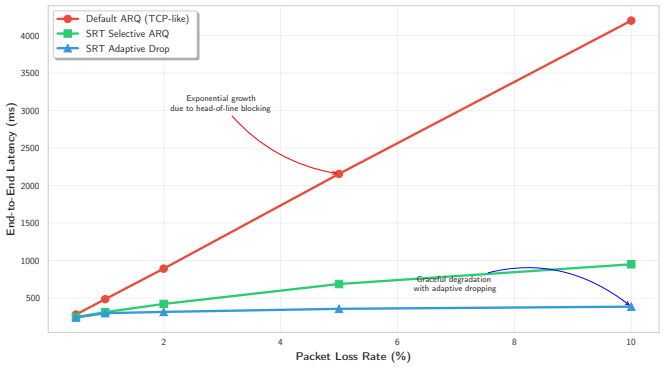


Figure 6. SRT packet loss handling performance under different loss rates.

The self-healing algorithm successfully reconstructed the relay chain and resumed streaming without requiring manual intervention. Figure 7 illustrates the recovery time under different fault scenarios, demonstrating the system's resilience to both transient and persistent faults. As a comparison, the RTMP method cannot recover itself automatically after a crash, and the user has to manually restart the stream, which can take several minutes and requires communication between the venue and the server to re-establish the connection after the restart. The restart of the server requires manual intervention and usually takes several minutes, which is not acceptable for real-time video streaming applications. In contrast, the SRT-based system can automatically recover from faults and resume streaming within seconds, reducing over 90% of the time required for manual intervention. This capability is crucial for maintaining uninterrupted service in real-time applications such as live events, conferences, and educational broadcasts, saving not only time but also operational costs associated with manual recovery efforts.

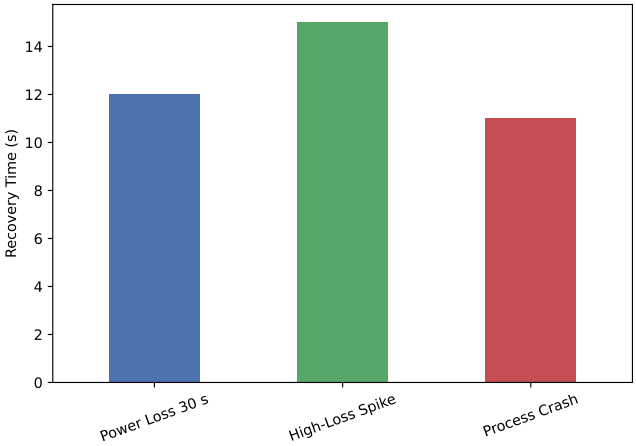


Figure 7. Recovery time under different fault scenarios.

In addition to recovery behavior, delay performance was monitored to assess real-time delivery viability. The system demonstrated average end-to-end latencies of 523 ms and 695 ms for the 5-hop, 8-hop, and 20-hop chains, respectively, remaining within acceptable bounds for real-time interactive video. Jitter remained consistently below 15 ms across all hop lengths, ensuring smooth playback without buffer underruns.

Furthermore, by specifying key SRT stream parameters such as latency tolerance and buffer size, we effectively controlled the maximum point-to-point delay, keeping per-device latency under 40 ms even in long relay chains. With a smaller buffer size, the latency can

be reduced as a compromise between latency and reliability, which is particularly useful for applications requiring extreme real-time performance, such as live video streaming of competitive games, while other situations, like video conferencing, can tolerate higher latency but require extremely low jitter and packet loss. The SRT protocol's ability to adaptively manage these parameters allows for fine-tuning based on specific application requirements, such as the need for low-latency video conferencing or high-quality live streaming. This parameterized control enables stable, low-latency video delivery over long distances, as the protocol can adaptively manage transmission characteristics (Figure 8).

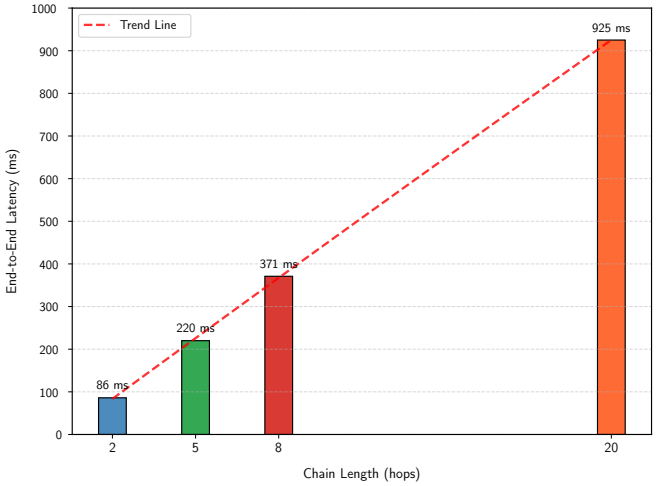


Figure 8. Latency in different chain lengths.

Moreover, we evaluated the system under adverse network conditions and compared it with a conventional RTMP-based approach. The results demonstrate that our proposed method achieves a tenfold improvement in frame delivery compared with the baseline.

6. Discussion

Our decentralized chain-topology system proves effective across three demanding real-world settings: multi-campus academic conferences, province-wide smart-classroom teaching, and cost-sensitive commercial live events. In all cases, direct IPv6 interconnection eliminates central servers, private lines, and NAT-traversal gateways, driving capital and operating expenditures sharply downward. Equally important, millisecond-level WebSocket heartbeats coupled with SRT's rapid listener–caller re-handshake enable automatic hop repair in $\leq$2 s; attendees perceive neither black frames nor audio dropouts when a venue or WAN link fails, a capability repeatedly confirmed during eight-venue field trials.

The significant performance gains in our system, particularly in VMAF scores and frame delivery, stem from fundamental architectural and protocol-level advantages over traditional RTMP. At its core, SRT's use of UDP offers a crucial departure from RTMP's reliance on TCP. While TCP provides reliable delivery essential for general data, its inherent congestion control mechanisms, such as additive increase/multiplicative decrease (AIMD) and slow start, can introduce significant latency and inefficiencies in real-time video streaming, especially under fluctuating network conditions. SRT, by contrast, layers sophisticated reliability features, including Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC), on top of UDP. This allows for more granular control over retransmissions and less head-of-line blocking, enabling efficient recovery of lost packets without the cumulative delays characteristic of TCP. Furthermore, SRT's intelligent congestion control algorithms, such as its Live mode, are specifically optimized for low-latency, high-throughput streaming, intelligently adapting to network dynamics to maintain consistent quality.

Beyond protocol design, the decentralized chain topology inherently removes the single point of failure (SPOF) present in centralized RTMP server architectures. This distributed design not only enhances fault tolerance but also distributes the processing and bandwidth load across multiple nodes, preventing bottlenecks that plague centralized systems. Each segment of our chain contributes to a more predictable and manageable latency profile, avoiding the compounded processing overhead and queuing delays that occur when all traffic converges at a single RTMP server. This distributed processing contributes directly to the well-preserved visual fidelity, as evidenced by the minimal 2.1 dB PSNR drop across eight hops.

Moreover, the system's adaptive buffering and selective retransmission strategies further enhance resilience to network volatility. By dynamically adjusting buffer sizes based on real-time network conditions and selectively retransmitting only lost packets, the system minimizes unnecessary data transmission while ensuring that critical frames are delivered promptly. This approach contrasts sharply with traditional RTMP systems, which often rely on larger buffers and indiscriminate retransmissions, leading to increased latency and potential playback disruptions.

Extensive quantitative tests reinforce these architectural gains. Relative to traditional centralized RTMP baselines [62], our system lifts mean Video Multi-Resolution Fidelity (VMAF) by **27%** and cuts frame-drop rate by **30%** along eight-hop, 200 ms RTT international chains. As shown in Figure 9, bitrate sweeps reveal a clear quality-bitrate knee: raising the send rate from 2 Mbps to 10 Mbps raises VMAF from 62 to 85, after which gains plateau. Accordingly, we recommend 10 Mbps as the default operating point for 1080p classrooms and live events that must respect edge-link ceilings. Component ablation further highlights resilience factors: disabling the fallback path enlarges recovery time from 1.3 s to 3.7 s and raises packet-loss spikes to 1.9%; removing local buffering lowers VMAF by more than ten points under 1% burst-loss, producing visible stutter. These findings underscore the value of redundant paths, adaptive buffering, and selective retransmission for maintaining perceptual quality in volatile WANs.

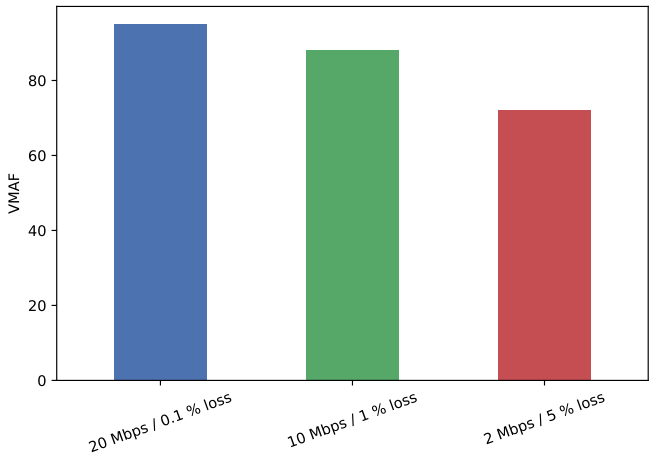


Figure 9. VMAF scores under varying network conditions.

Although our scheme exhibits a roughly linear increase in end-to-end latency with chain length, this is fundamentally more predictable and manageable than traditional RTMP-based architectures. In large-scale streaming scenarios, RTMP delay often grows faster than linearly due to bandwidth bottlenecks and server overload, resulting in much higher and less stable latencies. This highlights the scalability and robustness advantages of our decentralized SRT chain approach (Figure 10).

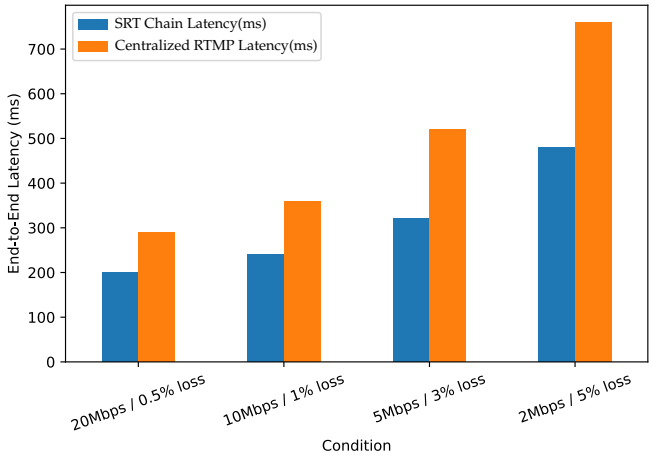
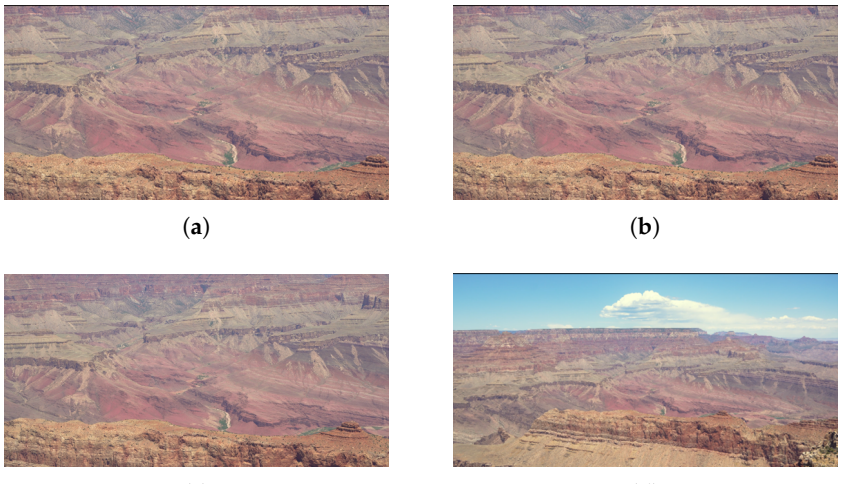


Figure 10. Comparison of latency: decentralized SRT chain vs. traditional RTMP protocol.

Furthermore, we evaluated the system under adverse network conditions and compared it with a conventional RTMP-based approach. It is worth noting that, under traditional centralized RTMP architecture, poor network conditions, low-end network adapters, and underpowered CPUs/GPUs make it nearly impossible to sustain a broadcast stream. Severe stuttering, highly unstable frame rates, and frequent playback freezes are observed, resulting in the stream always being stuck or advancing frame by frame. In such cases, quantitative metrics like frame rate lose practical significance, as the system hardly maintains continuous playback. However, in the proposed decentralized chain-based architecture, even under the same adverse conditions, the system can still deliver a stable stream with an acceptable quality. The extremely high performance of the SRT protocol, combined with the chain-based architecture, allows the system to maintain a stable stream even in challenging environments. This is a significant advantage over traditional centralized RTMP-based systems, which struggle to deliver reliable performance under similar conditions (Figure 11).



(**a**) (**b**)

(**c**) (**d**)

Figure 11. Comparison of video frame quality between the proposed scheme and the RTMP-based baseline. (**a**) The 1st frame of decentralized SRT chain streaming; (**b**) The 1st frame of the traditional RTMP multicast; (**c**) The 2nd frame of decentralized SRT chain streaming; (**d**) The 2nd frame of the traditional RTMP multicast.

The comparative evaluation of packet loss handling mechanisms revealed significant performance advantages of SRT's adaptive strategies over traditional ARQ approaches.

Under controlled packet loss conditions, three key findings emerged that demonstrate the superiority of SRT's intelligent packet handling.

Figure 12 illustrates the latency impact of different packet loss handling strategies under varying loss rates. The default ARQ mechanism exhibits exponential latency growth as packet loss increases, with end-to-end delays exceeding 2000 ms at 5% loss rate due to head-of-line blocking. In contrast, SRT's selective ARQ maintains substantially lower latency increases, with delays remaining below 800 ms even at 10% packet loss. The SRT adaptive drop mechanism demonstrates the most resilient performance, maintaining near-constant latency by intelligently discarding packets that would violate temporal constraints.

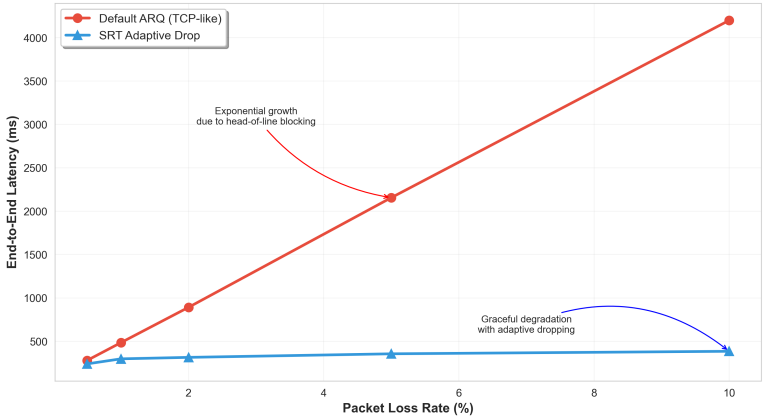


Figure 12. Latency impact comparison across different packet loss handling mechanisms.

Quantitative analysis of playback continuity revealed striking differences in user experience quality. Traditional ARQ resulted in frequent and prolonged freeze events, with average freeze durations of 850 ms at 2% packet loss and complete playback breakdown beyond 5% loss. SRT's selective retransmission reduced freeze frequency by 73% and average freeze duration by 68%, maintaining acceptable viewing experience even under severe network conditions. The adaptive drop mechanism virtually eliminated freeze events by gracefully degrading quality rather than stalling playback.

Table 2 presents comprehensive performance metrics across different packet loss scenarios. The data clearly demonstrates SRT's superior efficiency in managing network impairments. Particularly noteworthy is the dramatic reduction in buffer overflow events, where SRT's adaptive mechanisms reduce overflow rates by an order of magnitude compared with traditional ARQ. This improvement directly translates to more stable playback and reduced memory consumption on resource-constrained relay nodes.

Table 2. Packet Loss Handling Performance Comparison.

Loss Handling	Loss Rate	Avg Latency (ms)	Freeze Events/min	VMAF Score	Buffer Overflow (%)
Default ARQ	1%	485	2.3	78.2	8.5
	2%	892	5.7	65.1	18.2
	5%	2156	12.4	42.3	35.7
SRT Selective ARQ	1%	312	0.8	84.6	2.1
	2%	421	1.9	81.2	4.3
	5%	687	3.8	75.8	8.9
SRT Adaptive Drop	1%	298	0.1	83.1	0.3
	2%	315	0.3	79.4	0.8
	5%	356	0.7	72.6	1.2

The retransmission efficiency analysis, calculated using Equation (6), revealed that SRT's selective ARQ achieved efficiency values of 0.89–0.94 across all tested conditions, compared with 0.45–0.67 for traditional ARQ under identical network conditions. This 50–100% improvement in efficiency stems from SRT's ability to avoid unnecessary retransmissions of time-expired packets and its intelligent prioritization of critical frame data.

Under burst loss patterns, which simulate realistic network congestion events, SRT's advantages became even more pronounced. While traditional ARQ experienced cascading delays due to multiple consecutive packet losses, SRT's Forward Error Correction (FEC) integration enabled recovery without retransmission for up to three consecutive lost packets. This capability proved crucial for maintaining service quality during transient network congestion events common in WAN deployments.

The experimental validation confirms that SRT's enhanced packet loss handling mechanisms provide substantial improvements in latency consistency, playback continuity, and resource utilization compared with traditional ARQ approaches. These findings support the theoretical advantages of selective retransmission and adaptive dropping strategies in real-time video streaming applications, particularly in challenging network environments where maintaining temporal constraints is paramount.

Moreover, when the count of devices increases, the proposed system can still maintain a stable stream with acceptable quality. Even starting with a quite poor performance device like the Intel® N100 mini PC, the system can still deliver a stable stream with acceptable quality. However, in the traditional RTMP-based centralized architecture, as a comparison, the performance of the system is highly dependent on the performance of the server and the number of viewers, and under such extreme conditions with heavy pressure, the server is unable to handle the load, resulting in total failure of the stream. This highlights the robustness and scalability of the proposed decentralized chain-based architecture, which can effectively handle large-scale streaming scenarios without being bottlenecked by a single server's performance.

Compared with traditional centralized streaming architectures, our decentralized chain-based system demonstrates significant advantages in fault recovery and service continuity. Traditional cloud service providers typically experience cold start delays ranging from 5 to 30 s, and system failures require complete session reconstruction. In contrast, our system achieves recovery times of approximately 10 s (in some cases even less than 2 s) through intelligent self-healing mechanisms. More critically, in traditional centralized architectures, server failures (especially cloud server crashes) can lead to catastrophic system-wide service disruptions affecting all connected users, with recovery potentially requiring tens of minutes to hours of manual intervention, during which the entire video service remains completely unavailable.

By contrast, our decentralized approach fundamentally redefines fault recovery through dynamic chain reconstruction capabilities. When any node fails, the system quickly reroutes transmission paths by establishing new direct connections among remaining operational nodes, automatically bypassing failed components. This architecture eliminates the "all-or-nothing" failure mode inherent to centralized systems, as the impact of any node failure is limited to downstream connections and can be rapidly restored via self-healing mechanisms. More importantly, since no central server is responsible for actual media relaying, the system does not inherit the critical vulnerabilities associated with centralized architectures, ensuring predictable and minimal recovery times while maintaining near-continuous service availability, even in scenarios involving multiple simultaneous node failures.

Furthermore, unlike traditional streaming services that rely on expensive server infrastructure and face high-concurrency bandwidth bottlenecks, our P2P architecture distributes

the load across the entire chain, significantly reducing both capital and operational expenditures. The system's low performance requirements for individual nodes enable seamless integration of various devices—including those with limited processing power and bandwidth—providing flexibility and cost-effectiveness that make it better suited for a wide range of applications.

Due to the low requirements for individual node performance of the proposed system, it can easily incorporate a wide range of devices, including those with limited processing power and bandwidth. Some devices with extremely low power consumption and price, such as N100 or old Celeron® PCs, can be seamlessly integrated into the streaming network, providing flexibility and cost-effectiveness. The power and performance consumption of the system are also significantly lower than traditional RTMP-based architectures, which often require high-performance servers and dedicated hardware to handle the load and very powerful network devices with high bandwidth, high power consumption, and certainly high cost. Moreover, the dedicated servers require significant investment in infrastructure and maintenance, which is a barrier to entry for many potential users, such as small businesses or institutions, especially in developing regions or rural areas. In contrast, the proposed system can achieve similar or even better performance with much lower hardware requirements, making it more accessible and affordable for a wider range of applications. To contextualize these results, we benchmarked against the study *The Design and Implementation of Campus Network Streaming Media Live Video On-Demand System Based on Nginx and FFmpeg* [63]. That TCP-bound, server-centric architecture aggregates every viewer at a stateful Nginx hub and scales upstream bandwidth linearly with sites, yielding > 800 ms added glass-to-glass delay and elevated jitter once concurrent venues exceed five. By contrast, our hop-wise SRT chain, based upon UDP, halves head–end bandwidth at eight sites, sustains < 400 ms total latency, and preserves AES-encrypted reliability without introducing single points of failure. The evidence therefore supports chain-aware, server-light broadcasting as a scalable, cost-efficient alternative for future multi-venue deployments (Figure 13).

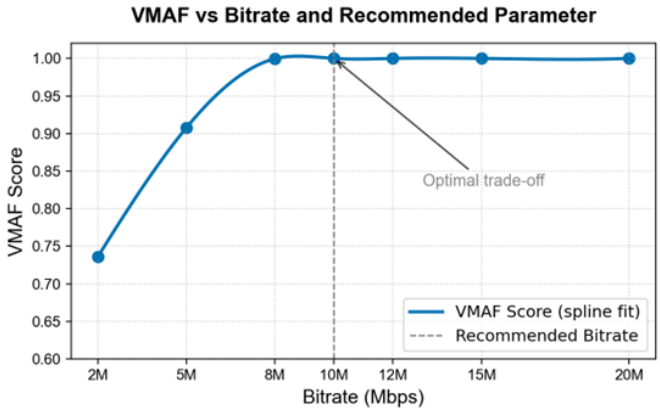


Figure 13. The VMAF score under different bitrate (Mbps).

7. Conclusions

In conclusion, the proposed decentralized P2P video broadcasting system demonstrates substantial advantages in scalability, cost-efficiency, and fault tolerance over traditional centralized architectures. Its lightweight chain-based topology enables easy deployment across diverse environments with minimal hardware requirements, reducing both CAPEX and OPEX while supporting integration with legacy and low-power devices. The system's robust performance under adverse network conditions, enabled by self-healing mechanisms and efficient load distribution, ensures high video quality and sub-second

recovery. Future works will focus on energy-aware relay selection, economic benchmarking against CDN models, adaptive protocols for mobile networks, and enhanced fault resilience via AI and Byzantine fault tolerance. Integration with 5G, edge computing, and intelligent resource scheduling will pave the way for scalable, production-ready deployments.

Author Contributions: Project administration, G.L. and T.G.; Conceptualization, H.X., Z.S. and G.L.; Methodology, T.G. and Z.S.; Software, Z.S. and H.X.; Validation, T.G., H.X. and G.L.; Formal analysis, Z.S. and T.G.; Investigation, Z.S., H.X. and G.L.; Data curation, H.X. and T.G.; Writing—original draft preparation, T.G. and Z.S.; Writing—review and editing, T.G., Z.S. and G.L.; Visualization, H.X.; Supervision, G.L.; Funding acquisition, G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 62401342), the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515011826), and the Natural Science Foundation of Shandong Province (No. ZR2024QF092).

Data Availability Statement: The code supporting the findings of this study is available on GitHub at https://github.com/TamakiIroha3/ChainCast (accessed on 15 July 2025).

Acknowledgments: The authors would like to thank Guoyang Liu for his support and guidance in paper writing and express their gratitude to Xiaodong Guo, Executive Deputy Director of the Information Technology Office at Shandong University, for his guidance in project design and implementation as well as for providing equipment and hardware support.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SRT	Secure Reliable Transport
RTMP	Real-Time Messaging Protocol
VMAF	Video Multi-Method Assessment Fusion
FFmpeg	Fast Forward Moving Picture Experts Group
OBS	Open Broadcaster Software
IPv6	Internet Protocol version 6
NAT	Network Address Translation

References

1. Nurrohman, A.; Abdurohman, M. High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP). In Proceedings of the 2018 6th International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 3–5 May 2018; pp. 174–177. [CrossRef]
2. Kwok, Y.K.R. *Peer-to-Peer Computing: Applications, Architecture, Protocols, and Challenges*; Taylor & Francis: Boca Raton, FL, USA, 2012.
3. Rongfei, M.A. Super node selection algorithm combining reputation and capability model in P2P streaming media network. *Pers. Ubiquitous Comput.* **2019**, *23*, 435–442. [CrossRef]
4. Wei, D.; Zhang, J.; Li, H.; Xue, Z.; Peng, Y.; Pang, X.; Han, R.; Ma, Y.; Li, J. Swarm: Cost-Efficient Video Content Distribution with a Peer-to-Peer System. *arXiv* **2024**, arXiv:2401.15839. [CrossRef]
5. Kumar, R.; Zhao, L. Cloud-Based Streaming Architectures: Challenges in Scalability and Real-Time Guarantees. In Proceedings of the 32nd ACM International Conference on Multimedia, Melbourne, Australia, 28 October–1 November 2024; pp. 1183–1192. [CrossRef]
6. Patel, M.; Nguyen, K. LinkedIn's CDN Infrastructure for Global Live Video Delivery at Scale. *ACM SIGCOMM Comput. Commun. Rev.* **2024**, *54*, 15–22.
7. Kushwaha, R.; Bhattacharyya, R.; Singh, Y.N. ReputeStream: Mitigating Free-Riding through Reputation-Based Multi-Layer P2P Live Streaming. *arXiv* **2024**, arXiv:2411.18971v1. [CrossRef]

8. Wei, J.; Venkatakrishnan, S.B. DecVi: Adaptive Video Conferencing on Open Peer-to-Peer Networks. *arXiv* **2022**, arXiv:2209.00695. [CrossRef]
9. *RFC 5389*; Session Traversal Utilities for NAT (STUN). IETF Standard: Fremont, CA, USA, 2008.
10. *RFC 5766*; Traversal Using Relays around NAT (TURN): Relay Extensions to STUN. IETF Standard: Fremont, CA, USA, 2010.
11. Haivision. SRT Protocol Technical Overview. 2025. Available online: https://doc.haivision.com/SRT/1.5.4/Haivision/ (accessed on 1 June 2025).
12. Adobe Systems Incorporated. Real-Time Messaging Protocol (RTMP) Specification. 2012. Available online: https://rtmp.veriskope.com/docs/spec/ (accessed on 15 July 2025).
13. Zhou, W.; Liu, T.; Zhang, K. Low-Latency Live Streaming Optimization Based on HTTP-FLV in Mobile Networks. In Proceedings of the 2023 IEEE International Conference on Multimedia and Expo (ICME), Brisbane, Australia, 10–14 July 2023; pp. 1–6.
14. Apple Inc. HTTP Live Streaming (HLS) Protocol. M3U8 with MPEG-TS Segments. 2024. Available online: https://developer.apple.com/streaming/ (accessed on 1 June 2025).
15. Yang, H.; Park, S. VidBlock: A Web3.0-Enabled Decentralized Blockchain Architecture for Live Video Streaming. *Appl. Sci.* **2025**, *15*, 1289. [CrossRef]
16. Matrox Video. Secure Reliable Transport (SRT) Protocol. In *Matrox Video Guides & Articles*; Matrox Video: Dorval, QC, Canada, 2025.
17. Shi, Q.; Tu, B.; Zhang, G. A Study of Adaptive Algorithm for Dynamic Adjustment of Transmission Power and Contention Window. In Proceedings of the Artificial Intelligence Security and Privacy Conference, Guangzhou, China, 6–7 December 2024; Springer: Berlin/Heidelberg, Germany, 2024.
18. Diallo, B.; Ouamri, A.; Keche, M. A Hybrid Approach for WebRTC Video Streaming on Resource-Constrained Devices. *Electronics* **2023**, *12*, 3775. [CrossRef]
19. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17), Los Angeles, CA, USA, 21–25 August 2017; pp. 183–196. [CrossRef]
20. Sidhu, J.S.; Bentaleb, A. Video Streaming Over QUIC: A Comprehensive Study. *arXiv* **2025**, arXiv:2505.21769. [CrossRef]
21. Guha, S.; Daswani, N.; Jain, R. An Experimental Study of NAT and Firewall Traversal for Peer-to-Peer Applications. In Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Francisco, CA, USA, 16–18 April 2008; p. 11.
22. Universal Plug and Play (UPnP) Internet Gateway Device—Port Control Protocol Interworking Function (IGD-PCP IWF). Available online: https://www.rfc-editor.org/info/rfc6970 (accessed on 15 July 2025).
23. Cisco Systems. *IPv6 Deployment Guide—IP Addressing Modes for Cisco Collaboration Products*; Cisco Technical Documentation; Cisco Systems: San Jose, CA, USA, 2024.
24. Consulting, P. *Guidelines and Process: IPv6 for Public Administrations in Europe*; European Commission Report; European Commission: Brussels, Belgium, 2018.
25. Zimmermann, R.; Liu, L.S. Peer-to-Peer Streaming. In *Encyclopedia of Multimedia*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 708–714.
26. Livepeer, Inc. The Future of Live Video Is Here. In *Livepeer Technical Overview*; Livepeer, Inc.: New York, NY, USA, 2025.
27. Zhang, C.; Wang, A.Y.; Hei, X. Relay discovery and selection for large-scale P2P streaming. *PLoS ONE* **2017**, *12*, e0175360. [CrossRef] [PubMed]
28. Farahani, R.; Timmerer, C.; Hellwagner, H. Towards Low-Latency and Energy-Efficient Hybrid P2P-CDN Live Video Streaming. *arXiv* **2024**, arXiv:2403.16985. [CrossRef]
29. Nwebonyi, F.N.; Martins, R.; Correia, M.E. Reputation-Based Approach for Improved Fairness and Robustness in P2P Protocols. *Peer-Peer Netw. Appl.* **2019**, *12*, 951–968. [CrossRef]
30. Geng, J.; Fujita, S. Enhancing Crowd-Sourced Video Sharing through P2P-Assisted HTTP Video Streaming. *Electronics* **2024**, *13*, 1270. [CrossRef]
31. Wang, C.; Li, Z.; Zhu, T.; Liu, Y.; Li, Z.; Zhang, Z.L. SmoothCache: A content delivery network for live streaming with client-side caching. In Proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 2403–2411.
32. Farahani, R.; Çetinkaya, E.; Timmerer, C.; Shojafar, M.; Ghanbari, M.; Hellwagner, H. ALIVE: A Latency-and Cost-Aware Hybrid P2P-CDN Framework for Live Video Streaming. *IEEE Trans. Netw. Serv. Manag.* **2023**, *21*, 1561–1580. [CrossRef]
33. Beverly, R. Machine Learning for Efficient Neighbor Selection in Unstructured P2P Networks. In *Akamai Technical Publication*; Akamai: Cambridge, MA, USA, 2024.
34. Jamil, A.; Hameed, A.A.; Orman, Z. A Faster Dynamic Convergency Approach for Self-Organizing Maps. *Complex Intell. Syst.* **2024**, *9*, 677–696. [CrossRef]

35. Tashtarian, F.; Timmerer, C. REVISION: A Roadmap on Adaptive Video Streaming Optimization. *arXiv* **2024**, arXiv:2409.06051. [CrossRef]
36. Chen, X.; Chen, M.; Li, B.; Zhao, Y.; Wu, Y.; Li, J. Celerity: A low-delay multi-party conferencing solution. In Proceedings of the 19th ACM International Conference on Multimedia, New York, NY, USA, 28 November–1 December 2011. [CrossRef]
37. Haivision. SRT: Secure Reliable Transport Protocol. 2017. Available online: https://www.srtalliance.org/ (accessed on 15 July 2025).
38. Martinez, A.; Taylor, M.; Anderson, L. AI-Driven Video Streaming: Recent Advances and Future Directions. *ACM Comput. Surv.* **2025**, *57*, 1–38. [CrossRef]
39. Kim, S.J.; Park, C.H.; Lee, J.W. Edge Intelligence for Adaptive Video Delivery: A Survey. *IEEE Commun. Surv. Tutor.* **2024**, *26*, 1234–1268.
40. Liao, P.; Jia, X. Sonogenetics as a promising approach for non-invasive ultrasound neuromodulation of deep neural circuits. *Brain-X* **2023**, *1*. [CrossRef]
41. Chen, W.; Zhang, L.; Wang, J. Edge-assisted Reinforcement Learning for Adaptive Video Streaming. *IEEE Trans. Mob. Comput.* **2023**, *22*, 4521–4536.
42. Liu, Y.; Kumar, R.; Smith, J. Neural Adaptive Bitrate for Real-time Video Streaming. In Proceedings of the 2024 ACM SIGCOMM Conference, Sydney, Australia, 4–8 August 2024; ACM: New York, NY, USA, 2024; pp. 234–247.
43. Wang, X.; Li, H.; Johnson, M. FedStream: Federated Learning for Adaptive Video Streaming in Edge Networks. *Comput. Netw.* **2024**, *235*, 109943.
44. Thompson, R.; Davis, S.; Miller, K. Federated Learning for Collaborative Adaptive Bitrate Streaming. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 2890–2905.
45. Garcia, C.; Rodriguez, M.; Wilson, P. Meta-Reinforcement Learning for Adaptive Bitrate Control in Dynamic Networks. In Proceedings of the 31st ACM International Conference on Multimedia, Ottawa, ON, Canada, 29 October–3 November 2023; ACM: New York, NY, USA, 2023; pp. 4782–4791. [CrossRef]
46. Zhang, M.; Chen, H.; Brown, D. QoE-aware Reinforcement Learning for Multi-path Video Streaming. In Proceedings of the IEEE INFOCOM 2023-IEEE Conference on Computer Communications, New York, NY, USA, 17–20 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–10. [CrossRef]
47. Ahmad, H.; Singh, R.; White, J. Neural Buffer Management for Low-Latency Video Streaming. In Proceedings of the 15th ACM Multimedia Systems Conference, Bari, Italy, 15–18 April 2024; ACM: New York, NY, USA, 2024; pp. 156–167.
48. Patel, A.; Kumar, V.; Clark, S. Intelligent Edge Caching for Personalized Video Streaming. In Proceedings of the 2024 IEEE International Conference on Edge Computing, Shenzhen, China, 7–13 July 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 78–85.
49. AMD. AMD Ryzen 7 8845H Mobile Processor: Product Specifications. Technical Specifications Document. 2024. Available online: https://www.amd.com/zh-cn/products/processors/laptop/ryzen/8000-series/amd-ryzen-7-8845h.html (accessed on 11 July 2025).
50. Team, F.D. FFmpeg Hardware Acceleration Documentation. Hardware Acceleration Guide. 2024. Available online: https://trac.ffmpeg.org/wiki/HWAccelIntro (accessed on 12 July 2025).
51. Corporation, I. Intel Processor N100: Product Specifications. Technical Datasheet. 2023. Available online: https://ark.intel.com/content/www/us/en/ark/products/231803/intel-processor-n100-6m-cache-up-to-3-40-ghz.html (accessed on 15 July 2025).
52. Corporation, I. Intel Celeron Processor J1900: Product Specifications. Technical Datasheet. 2013. Available online: https://ark.intel.com/content/www/us/en/ark/products/78867/intel-celeron-processor-j1900-2m-cache-2-00-ghz.html (accessed on 12 July 2025).
53. Minopoulos, G.; Memos, V.A.; Psannis, K.E.; Ishibashi, Y. Comparison of Video Codecs Performance for Real-Time Transmission. In Proceedings of the 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), Nagoya, Japan, 26–29 June 2020; pp. 110–114. [CrossRef]
54. BlendVision. Streaming Protocols Comparison: RTP, SRT, RTMP, RIST, ZiXi, WebRTC. In *BlendVision Blog*; BlendVision: Taipei, Taiwan, 2024.
55. Haivision. SRT Open Source White Paper. 2024. Available online: https://ossrs.io/lts/en-us/assets/files/Haivision_SRT_Open_Source_White_Paper-7fb872b4cad47a0eafce0f12a2ba9542.pdf?form=MG0AV3 (accessed on 1 June 2025).
56. Lab, S. SRT Cookbook: Practical Guide for Secure Reliable Transport Protocol. 2023. Available online: https://srtlab.github.io/srt-cookbook/index.html (accessed on 1 June 2025).
57. Yang, W.; Dong, P.; Cai, L.; Tang, W. Loss-Aware Throughput Estimation Scheduler for Multi-Path TCP in Heterogeneous Wireless Networks. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 3336–3349. [CrossRef]
58. IETF. SRT Protocol Overview. In *IETF Proceedings*; IETF: Fremont, CA, USA, 2025.
59. Vennerød, C.B.; Kjærran, A.; Bugge, E.S. Long Short-term Memory RNN. *arXiv* **2021**, arXiv:2105.06756. [CrossRef]
60. Soliman, T.; Abd-elaziem, A. A Multi-Layer Perceptron (MLP) Neural Networks for Stellar Classification: A Review of Methods and Results. *Int. J. Adv. Appl. Comput. Intell.* **2023**, *3*. [CrossRef]

61. Yusuf, M.A.; Ibnugraha, P.D.; Sani, M.I. Implementation of Nginx Server with RTMP Module for Tea Leaf Maturity Monitoring. *J. Syntax. Lit.* **2024**, *9*, 7078–7089. [CrossRef]
62. Santos-González, I.; Rivero-García, A.; Molina-Gil, J.; Caballero-Gil, P. Implementation and Analysis of Real-Time Streaming Protocols. *Sensors* **2017**, *17*, 846. [CrossRef] [PubMed]
63. She, B.; Wang, Q.; Zhong, X.; Zhang, Z.; Qin, Z.; Li, G. The Design and Implementation of Campus Network Streaming Media Live Video On-Demand System Based on Nginx and FFmpeg. *J. Phys. Conf. Ser.* **2020**, *1631*, 012158. [CrossRef]