DSJA: Distributed Server-driven Joint Route Scheduling and Streaming Adaptation for Multi-Party Realtime Video Streaming

Journal:	*Transactions on Mobile Computing*
Manuscript ID	TMC-2023-08-0907
Manuscript Type:	Regular
Keywords:	J.9.c Multimedia applications and multimedia signal processing < J.9 Mobile Applications < J Computer Applications, C.2.8.a Algorithm/protocol design and analysis < C.2.8 Mobile Computing < C.2 Communication/Networking and Information Technolog, M.4.4.h Quality of Services < M.4.4 Solution Reference Architectures < M.4 Service-Oriented Architecture < M Services Computing

DSJA: Distributed Server-driven Joint Route Scheduling and Streaming Adaptation for Multi-Party Realtime Video Streaming

Dayou Zhang *Student Member, IEEE*, Hao Zhu, Kai Shen, Dan Wang *Senior Member, IEEE*, Fangxin Wang *Member, IEEE*

Abstract—The widespread availability of convenient wireless network connection and video capture have fueled the development of multi-party realtime video streaming (MRVS) services, such as Zoom or Microsoft Teams. These services have transformed the generation and distribution of realtime streaming content and offer a new way of online communication, striving to provide high Quality-of-Experience (QoE) for individuals. However, delivering high QoE in MRVS is more challenging than in traditional video scenarios due to the stringent delay requirements and complex multi-party interactive architectures. In this paper, we propose DSJA, a distributed server-driven multi-party realtime video streaming framework that conquers the challenges. We first design an appropriate QoE model for MRVS services to capture the interplay among perceptual quality, variations, bitrate mismatch, loss damage, and streaming delay. We then model the QoE maximization problem in MRVS as a route scheduling and streaming adaptation problem. Afterward, we design DSJA which seamlessly integrates multiple selective forwarding units (SFU) architecture and server-driven approaches based on a two-step solution. Our evaluations show that our framework outperforms state-of-the-art solutions by $23.1\% \sim 41.7\%$ from the perspective of QoE, and reduces the backbone network transmission by $14.0\% \sim 36.6\%$.

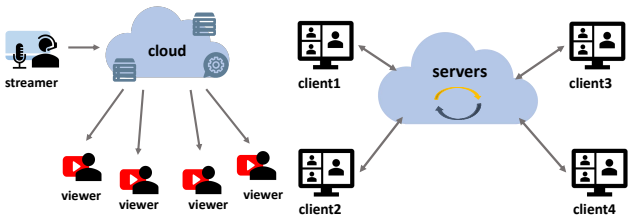
Index Terms—Multi-party realtime video streaming, Adaptive bitrate control, Quality of Experience

1 Introduction

Multi-party realtime video streaming (MRVS) has become increasingly popular as a communication method, allowing users worldwide to share their video calls in realtime. MRVS applications, such as video conferencing, online gaming, and telemedicine, have witnessed a surge in popularity, resulting in significant internet traffic. Distinct from Video-on-Demand (VoD) services [2], such as YouTube, and crowd-sourced live streaming services, like Twitch (see Fig. 1(a)), multi-party realtime video streaming or conferencing (see Fig. 1(b)) represents a new form of communication where each client can transmit its video stream to all other participants while simultaneously receiving streams from others in realtime. In the foreseeable future, MRVS is expected to continue generating substantial revenue. According to Sandvine's Global Internet Phenomena Report [3], video content has accounted for 65% of all data traffic and 69% of all mobile data traffic. Realtime video streaming traffic has comprised 17% of total video traffic and continues to grow at a rapid pace [4].

Ensuring a satisfactory Quality of Experience (QoE) for users is critical in MRVS; however, the optimal QoE is hard to achieve due to the new challenges in MRVS. *The first one lies in the packet loss problem caused by the unreliable transmission in MRVS, which can further lead to delay increase and visual quality degradation if without careful processing.* Unlike TCP-based reliable transmission (e.g., VoD and live streaming), MRVS usually employs lightweight, yet unreliable, transmission protocols based on UDP (or its variants in conjunction with control protocols [5]) to maintain fast responses. When packet loss occurs, current systems either wait for retransmission, increasing delay, or directly deliver the incomplete stream to upper applications, potentially distorting video quality.

The second challenge arises from the difficulty of scheduling limited network resources amid the transformation from regional two-party communication to global multi-party communication. While traditional VoD or live streaming services use a client-server architecture where each client only manages communication with the other party, MRVS involves a more complex network topology with multiple clients and potential servers. This complexity gives rise to new issues such as selective forwarding units (SFU) selection, route path scheduling, and stream orchestration based on requests.



(a) Live Streaming (b) Video Conferencing

Fig. 1. The representative applications of realtime streaming services.

- *A preliminary version of this work [1] appeared in IEEE INFOCOM 2023.*
- *Dayou Zhang, Hao Zhu, Kai Shen, and Fangxin Wang are all with the SSE and FNii, The Chinese University of Hong Kong, Shenzhen. (email: {dayouzhang,haozhu,kaishen}@link.cuhk.edu.cn, {wangfangxin}@cuhk.edu.cn)*
- *Dan Wang is with The Hong Kong Polytechnic University, Hong Kong (e-mail: csdwang@comp.polyu.edu.hk).*
- *Dayou Zhang and Hao Zhu contributed equally to this work.*

Manuscript received xxx; revised xxx.

Previous works on bitrate adaptation [6]–[9] only addressed two-party communication scenarios, leaving a significant gap in fine-grained resource allocation for higher user QoE in multi-party cases.

The third challenge is how to coordinate the transmission behaviors of different parties so as to maximize the overall QoE of a communication group. Conventional works [10]–[13] mostly start from a client-driven perspective and only make local decisions. Such an independent decision-making strategy can easily cause poor QoE from a global view. However, it is not easy to coordinate the interests of multiple parties, especially when considering a variety of impact factors such as loss, bitrate, and other network conditions.

The fourth challenge pertains to the complex system architecture involving distributed SFUs. Traditional CDN-based video services are typically structured in a star topology with a single virtual or physical server. Such an architecture underutilizes the capabilities of modern mesh topology backbone networks, particularly when there are numerous and geographically dispersed users. However, implementing a distributed multi-SFUs server setup is not a straightforward task. It requires careful coordination of the potential SFUs' capacities and the identification of the best routing path.

Pioneer research works have made efforts toward these challenges. Sun et al. [14] developed Deep Reinforcement Learning frameworks to ensure low end-to-end video latency in live streaming services, but lack consideration for packet loss problems. Zhang et al. [15] proposed a joint bitrate and loss adaptation scheme for realtime video streaming, while they only focused on two-party communication. MultiLive [16] addressed the bitrate allocation in a multi-party live streaming scenario, but it only considered reliable transmissions. Therefore, existing works only partially tackle the challenges therein, calling for an integrated solution to address these crucial problems in delay, multi-stream orchestration, and multi-party coordination towards QoE maximization.

In this paper, we propose the DSJA framework, a framework for multi-party realtime video streaming services that can address the challenges and maximize user QoE. The coordination of DSJA focuses on two parts: (1) Route scheduling: the route scheduler considers the current SFU's capacity and network conditions to guide clients in setting up connections. The routing will affect the fundamental video latency, which is a significant factor of users' QoE. (2) Global streaming adaptation: the stream controller conducts joint loss and bitrate adaptation decisions with global orchestration based on network conditions during streaming. Following the server-side decision, the senders conduct a globally optimal stream configuration with Scalable Video Coding (SVC), and the server forwards the optimal stream layer to corresponding receivers. To the best of our knowledge, DSJA is the first server-driven joint loss and bitrate adaptation MRVS framework and the first multi-SFUs route scheduling MRVS framework.

Real-world network datasets, including the topology of the North America Backbone network and network traces of different network conditions, are used to evaluate DSJA's performance. The results demonstrate that DSJA effectively improves the average QoE by $23.1\% \sim 41.7\%$ compared to state-of-the-art existing approaches. The multi-SFUs design of DSJA also reduces backbone network pressure by transmitting $14.0\% \sim 36.6\%$ less data than other baselines.

TABLE 1
Service Types of Pioneer Research

	realtime	multi-party	loss adaptation	route scheduling	global coordination
DRL-Live [14]	✓				
Oppugno [15]	✓		✓		
vSkyConf [17]		✓			✓
AgRank [18]	✓	✓			
MultiLive [16]	✓	✓			✓
DSJA	✓	✓	✓	✓	✓

The remainder of the paper is organized as follows. Section 2 introduces the motivation of our framework. Section 3 illustrates the framework architecture. Section 4 formulates the problem with the QoE model definition. Section 5 illustrates our solution to the route scheduling problem and Section 6 describes the design of the streaming adaptation algorithm. We present the simulation results through extensive trace-driven experiments in Section 7. Section 8 discusses the related work. Finally, we conclude the paper in Section 9.

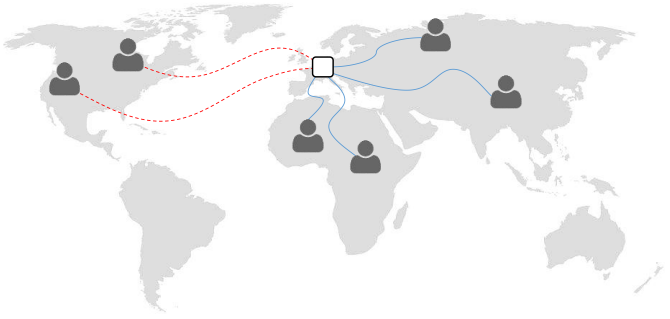
2 Motivation

Researchers have become increasingly interested in multi-party realtime video streaming services given their great market value. Different from VoD or live services, MRVS has several key features: *1) realtime*: an acceptable transmission delay is usually below 150 ms [19], far less than VoD and live services; *2) simultaneous transceiving*: each node can send messages while at the same time receiving messages from others, e.g., in a video conference; *3) multi-party*: multiple nodes can have all-to-all communication; *4) unreliable transmission*: existing systems mostly employ UDP-based protocol, which cannot guarantee the reliability in order to satisfy fast response. *5) complex topology*: MRVS have widespread users that utilize the MESH-based complex real-world network, with the usage of multiple potential SFUs.

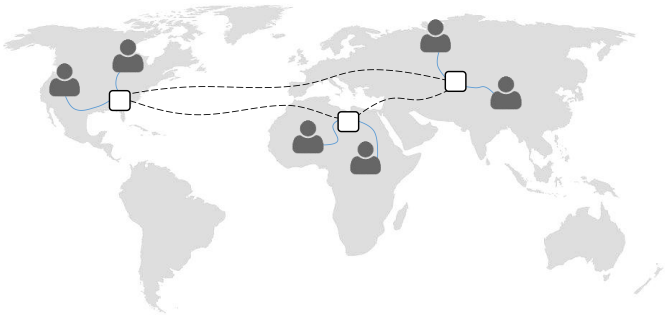
The confluence of these features makes the MRVS service provision even more challenging towards maximized user QoE, calling for an integrated framework that jointly considers realtime, multi-party, loss adaptation, route scheduling, and global coordination for transmission optimization. Pioneer works [14]–[18] have made attempts to address this problem. However, as summarized in Table 1, they only consider partial factors and fail to achieve a comprehensive solution. Some other related works try to minimize the global network cost [20], reduce the end-to-end delay [21], or meet the fairness constraints [22], which also only tackle partial problems.

We have closely examined the features of MRVS services and summarized three insights as follows.

Firstly, we have found that the multiple SFUs architecture outperforms other architectures in MRVS. One of the most significant factors affecting users' QoE in MRVS is video



(a) Single SFU Architecture



(b) Multi-SFUs Architecture

Fig. 2. Comparison of single SFU or multiple SFUs.

(a) 480p video with 1% loss (b) 480p video with 2% loss

(c) 1080p video with 1% loss (d) 1080p video with 2% loss

Fig. 3. Visual effects when multiple viewers experience different loss rates.

latency between clients. The streaming latency is determined by the network condition and geographical distance of the routing paths between clients. However, the single SFU architecture may increase such latency, especially when clients are far from each other. As depicted in Fig. 2(a), regardless of where the single SFU is located, clients on the same continents must send their video data across the continents to reach each other through the only SFU. A much better solution is demonstrated in Fig. 2(b) where each client connects to a closer SFU, which can then utilize the backbone network more efficiently.

Secondly, we argue that extra bandwidth can be leveraged for redundant coding such that loss can be corrected without retransmission. As shown in Fig. 3, even 1% of bit errors in the transmission can gravely damage viewers' perceptions. Even worse, in multi-party conferencing, if bit errors occur during video uploading, the user experience of all viewers at different bitrate levels will be degraded. As a result, the impact of network loss problems induced by fast yet unreliable UDP should not be underestimated. Therefore, Forward Error Correction (FEC) mechanisms can be utilized to mask the effect of loss, where the extra bandwidth can be leveraged to facilitate loss recovery.

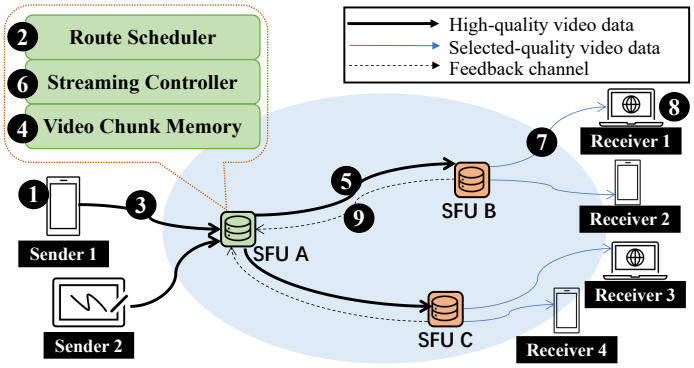


Fig. 4. Framework architecture of DSJA.

Thirdly, we emphasize that the overall QoE maximization of MRVS can only be achieved through server-driven approaches. Studies towards QoE maximization usually focus on bitrate adaptation, where an ABR controller will be responsible for selecting the most appropriate bitrate based on the available network throughput [11], [23]–[26], receiver buffer occupancy [26]–[28], or utilize reinforcement learning approaches [10], [29]. These client-driven approaches only utilize local state information to request their own most suitable bitrates, while the decisions are usually not globally optimal in such a non-cooperative way. For example, if multiple viewers request the same video with different bitrates from a sender, it will increase the sender encoding time, resulting in higher latency for all receivers and failing to achieve satisfactory overall QoE.

Thus, motivated by the importance of route scheduling, the non-negligible loss damages, and the necessity of server-driven approaches, we take it one step further and explore, for the first time, the effects of distributed server-driven joint route scheduling and stream adaptation in MRVS.

3 Framework Overview

The DSJA framework of distributed server-driven joint route scheduling and stream adaptation in MRVS aims to maximize overall QoE. As illustrated in Fig. 4, DSJA follows a server-driven architecture responsible for coordinating the streaming configuration of all clients. It mainly consists of five components, i.e., sender, receiver, route scheduler, server controller, and video chunk memory. It is worth noting that different from the logical distinction, a client in MRVS service can serve as both sender and receiver at the same time. The detailed functionalities are as follows:

- **Sender:** A sender is responsible for sending out streams with different bitrates employing scalable video coding [30]. SVC is an extension to the H.264/AVC [31]

video codec standard, which can encode a video stream into a base layer and several enhancement layers. Besides, the sender also leverages redundant coding techniques (i.e., FEC [32]) in our design, aiming to provide error recovery ability to avoid retransmission.

- **Receiver:** Since the receiver has all the local updated information, such as buffer occupancy and throughput estimation, it first conducts an initial request for bitrate using ABR algorithms, which have been widely explored by pioneer works [2], [12]. In order to support global coordination, the feedback information such as the receiver's QoE, bitrate decision, and network conditions needs to be sent back to the server for further control.
- **Route Scheduler:** The route scheduler stores information on different video sessions and is responsible for route scheduling and SFUs selection. When a user attempts to join a video session, they first connect to the route scheduler in the backbone network. The scheduler then calculates the most suitable SFU and streaming routes for the video session based on SFUs' job queuing delay and path latency through our route scheduling algorithm (§5). Following the result, clients can establish a video stream with the selected SFUs.
- **Stream Controller:** The server control unit is the core component in the cloud server responsible for global coordination. It collects the state information from senders and receivers, as well as the receiver feedback information. Through our designed streaming adaptation algorithm (§6), it derives the optimal bitrate and loss adaptation decisions from a global view and delivers the control signals to the senders for the next chunk streaming.
- **Video Chunk Memory:** Similar to a common SFU architecture, the video chunk memory at the cloud server will collect streams from multiple senders, conduct stream orchestration, and then forward corresponding streams with appropriate bitrate to different receivers based on the decision result. Thanks to the SVC format, the video chunk memory is able to conduct lightweight layered streaming without much transcoding overhead.

Our framework operates every time cycle, which is a short time slot. The workflow of a decision cycle is demonstrated in Fig. 4. Each sender (❶) first finds the most suitable SFU by querying the route scheduler(❷). Then the sender generates an uplink video stream (❸) with multiple bitrate layers to the video chunk memory (❹) in the SFU. The SFU will distribute the high-quality video data to other related SFUs through the backbone network(❺). After receiving the control signals from the stream controller (❻), the downlink streams (❼) with different layered bitrates and different coding redundancy will be forwarded to corresponding receivers (❽). At the end of each time slot, the server control unit will send network information to other SFUs through the feedback channel.

4 QoE Model and Problem Formulation

We mathematically model the problem in this section to better understand the challenges under MRVS scenarios. To do so, we introduce our QoE design, which is the ultimate objective, followed by constructing the route scheduling problem and streaming adaptation problem formulation and explaining the network constraints.

We assume that there are I senders and J receivers in the multi-party realtime video streaming service, where $\mathbf{I} = \{1, 2, ..., I\}$ represents the set of senders and $\mathbf{J} = \{1, 2, ..., J\}$ represents the set of receivers. Note that I may not equal J. Also, there are $\mathbf{R} = \{R_1, R_2, ..., R_q\}$ types of encoding bitrate levels and $\mathbf{C} = \{C_1, C_2, ..., C_p\}$ types of FEC code rate levels. Additionally, there are K timeslots, where $\mathbf{K} = \{0, 1, 2, ..., K\}$ represents the set of slotted time indexes.

4.1 QoE Model

Followed by existing works [33], [34], we consider five QoE metrics, i.e., *video rate based perceptual quality*, *perceptual quality variations*, *bitrate mismatch level*, *streaming delay* and *loss penalty*, where we separate the QoE degradation caused by bit errors from the original perceptual quality related to bitrate. Firstly, we can calculate the perceptual quality of sender i's video in receiver j at time slot k as follows:

$$q(r_k^{i,j}) = \log(\frac{r_k^{i,j}}{r_{min}}) \tag{1}$$

where $r_k^{i,j}$ denotes the bitrate level from sender i to receiver j at slotted time k, and r_{min} denotes the minimal bitrate level. Here we adopt the logarithmic function [28] to represent the video quality, where the ratio of real bitrate to minimal bitrate can be utilized to decrease the marginal quality improvement.

Secondly, we take advantage of the perceptual quality variations to penalize changes in video quality to favor smoothness [2]. In more detail, we adopt the absolute value of the difference between current video quality and the video quality of the last moment to represent the variations as follows:

$$\begin{aligned} |q(r_k^{i,j}) - q(r_{k-1}^{i,j})| &= |\log(\frac{r_k^{i,j}}{r_{min}}) - \log(\frac{r_{k-1}^{i,j}}{r_{min}})| \\ &= |\log(\frac{r_k^{i,j}}{r_{k-1}^{i,j}})| \end{aligned} \tag{2}$$

Thirdly, we argue that the assigned streaming may differ from the requested streaming of the receiver's ABR controller in practice. Thus, the bitrate mismatch penalty demonstrates the dissatisfaction between the receivers' requested bitrate and the transmitted bitrate. We can calculate it as follows:

$$B(r_k^{i,j}, \hat{r}_k^{i,j}) = \log(\frac{\hat{r}_k^{i,j}}{r_{min}}) - \log(\frac{r_k^{i,j}}{r_{min}}) = \log(\frac{\hat{r}_k^{i,j}}{r_k^{i,j}}) \tag{3}$$

Fourthly, the following components contribute to the metric of streaming delay: 1). the link latency which is associated with the processing time of the SFU and the transport path between sender and receiver. 2). the video encoding latency, which is proportional to the bitrate selected. 3). the uploading latency of the sender to upload the video stream. 4). the downloading latency of the receiver to download the video stream. The metric is shown as:

TABLE 2
Notations used in this paper

Variables	Meaning
$r_k^{i,j}$	real bitrate from sender i to receiver j at time k
r_{min}	minimal bitrate level
$\hat{r}_k^{i,j}$	requested bitrate of sender i from receiver j at time k
$d_{i,j}(k)$	streaming delay between sender i and receiver j
c_k^i	FEC code rate of sender i at time k
$\mathbf{r}_k^i$	the set of all bitrates encoded by sender i at time k
$T(\mathbf{r}_k^i)$	encoding time of sender i at time k
$B_i^{up}(k)$	uplink bandwidth of sender i at time k
$B_j^{down}(k)$	downlink bandwidth of receiver j at time k
$p_i(k)$	loss rate between client i and the corresponding SFU at time k
$e_{i,j}(k)$	remaining loss rate from sender i to receiver j at time k
T	duration of each time slot

$$d_{i,j}(k) = D_{i,j} + T(\mathbf{r}_k^i) + \frac{\max_j r_k^{i,j}}{c_k^i \cdot B_i^{up}(k)} + \frac{\sum_{i=1}^{I} \frac{r_k^{i,j}}{c_k^i}}{B_j^{down}(k)} \quad (4)$$

In the expression, the link latency Di, j equals the base route latency of the route between user i to user j plus the job queuing time which is associated with the computational capacity of corresponding SFU to user i. $\mathbf{r}_k^i = \{r_k^{i,j} | r_k^{i,j} \in \mathbf{R}, j \in \mathbf{J}\}$ represents the set of all bitrates that sender i will encode at time k. Note that c_k^i is the FEC code rate, equal to k/n, where an encoder takes a block of k source symbols as inputs and generates a total of n FEC symbols ($n > k$) as output. It is notable that our design is based on an asynchronous model that uses past feedback to make decisions in the coming time slot. Therefore, the decision-making time and feedback transmission time are ignored.

Lastly, we consider the QoE degradation caused by bit errors. We denote $e_{i,j}(k) = \max\{p_i(k) + p_j(k) - (1 - c_k^i), 0\}$ to represent the remaining bit error rate after the FEC recovery, and we map the bit errors to the QoE degradation penalty through $L(e_{i,j}(k))$, which will be analyzed in later section.

Therefore, we have considered all five QoE metrics in multi-party realtime video streaming. In conclusion, the QoE for receiver j at slotted time k is defined as:

$$\begin{aligned} QoE_j(k) = \sum_{i=1}^{I} \alpha_j^i \{ & \beta_j q(r_k^{i,j}) - \gamma_j | q(r_k^{i,j}) - q(r_{k-1}^{i,j}) | \\ & - \delta_j B(r_k^{i,j}, \hat{r}_k^{i,j}) - \epsilon_j d_{i,j}(k) - \zeta_j L(e_{i,j}(k)) \} \end{aligned} \quad (5)$$

where α_j^i is the importance factor of receiver j towards the video from sender i, and other Greek letters represent the receiver's personalized viewing demand.

Given the previous definition, several decisions will affect the user's QoE, including SFUs and paths selection, the bitrate and FEC rate when uploading to the SFU, and the requested bitrate when downloading from the SFU. **We can divide this QoE optimization problem into two parts: the route scheduling problem and the streaming parameter adaptation problem.** The route planning problem determines link paths and active SFUs of video streaming sessions which will significantly influence video latency. The encoding-related problem infers the bitrate and FEC rate and is responsible for users' QoE.

4.2 Route Scheduling Problem Modelling

In the routing problem, we need to find the optimal routes between clients and servers that are far distributed. The sender uploads the video stream to a server, which is then forwarded through other servers before reaching the receivers by different bitrate. Due to the computational and throughput capacity of the server, and transmission delays over the links, this process forms the fundamental video streaming latency. The objective of this model is to minimize the fundamental communication delay between clients from a network routing perspective.

We can model this problem as a graph theory problem. In the graph, vertices consist of clients, servers, and routers. Each server has a vertex weight representing its processing delay, while clients and routers are weighted 0. Edges represent the route between vertices, and the weight of an edge represents the communication delay between two vertices. Our objective is to find the minimum total cost between each pair of clients. We can use the following model to describe this problem:

Let $G = (V, E)$ be an undirected graph, where V represents the set of vertices and E represents the set of edges. Let U denote the set of clients, S denote the set of servers, L denote the set of routers, w_i represent the vertex weight of vertex i, and $c_{i,j}$ represent the edge weight of edge (i, j). Our objective is to select a subset of vertices and edges, denoted as $G' \subseteq G$, such that all clients in U are connected within G', and the total pairwise distance between clients in G' is minimized. It is important to note that the resulting graph G' may contain cycles in general. However, we impose the constraint that any vertices within these cycles must not include points belonging to the set U. This restriction is necessary due to the limited bandwidth of clients, ensuring that clients are not part of a cycle that forwards streams for other clients.

4.3 Stream Adaptation Problem Formulation

Integrating the viewers' personalized QoE demands and overall viewing architecture, we have the following optimization objective that aims to maximize the overall QoE driven by the server end under multi-party realtime video streaming:

$$\begin{aligned} \max_{\{r_k^{i,j}, c_k^i\}} \quad & \frac{1}{K} \sum_{k=0}^{K-1} \left[\sum_{j=1}^{J} \eta_j \cdot QoE_j(k) \right] \\ s.t. \quad & \max_j \{ \tfrac{r_k^{i,j}}{c_k^i} \} \leq B_i^{up}(k), \forall i \in \mathbf{I} \\ & \sum_{i=1}^{I} \tfrac{r_k^{i,j}}{c_k^i} \leq B_j^{down}(k), \forall j \in \mathbf{J} \\ & \tfrac{1}{K} \sum_{k=0}^{K-1} d_{i,j}(k) \leq \lambda T, \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \\ & r_k^{i,j} \in \mathbf{R}, c_k^i \in \mathbf{C} \end{aligned} \quad (6)$$

where T is the duration of each time slot, and η_j represents the importance of receiver j in the meeting room. Our action space consists of the actual bitrate of each sender-receiver pair and the FEC code rate of every sender. The first constraint is the upload bandwidth constraint for each sender, and the second is the download bandwidth constraint for every receiver. The third constraint is the time-average expected delay requirement, which ensures the low

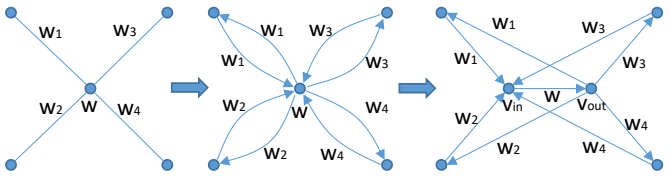


Fig. 5. Graph transformation.

latency of the service. The last one guarantees the available bitrate and FEC code rates, which belongs to pre-setted combinations and are non-negative.

5 Route Scheduling

In this section, we first reconstruct the graph problem into a more general version by reassigning the point weight to the edge weight. We then propose our relaxation and algorithm to the route scheduling problem.

5.1 Graph Reconstruction

In Section 4.2, we model our route problem into a graph problem. However, this graph contains nodes with associated weights, which can pose challenges for many graph algorithms. Therefore, we first transform the weighted vertex into two unweighted vertices with a weighted edge.

Fig. 5 shows the detail of the graph transformation. It first involves transferring the undirected graph into a directed graph. Then, we splitting each node with weight w into two nodes, labeled v_{in} and v_{out}, connected by an edge (v_{in}, v_{out}) with weight w. The original edges incident to the weighted node v are split into two edges, one connecting v_{in} to the neighbor node u and another connecting v_{out} to u. Another trick here is that we only connect the v_{out} between SFUs. This is because only the job queuing cost of the first SFU needs to be calculated, while other SFUs only do the stream forwarding.

After these two transformations, the cost on each path equals the sum of the edges of the path. Therefore, the routing planning problem is fitted into the traditional graph problem.

5.2 NP-hardness and Proof

Theorem 1. The Routing Scheduling Problem in Sec. 4.2 is NP-hard.

Proof: Based on the definition provided in Section 4.2, a general target graph G' can contain cycles, but any vertices within these cycles must not include points belonging to the set U. However, we consider a special case of our problem where both set S and L are empty. In this scenario, the vertex set V is equivalent to the client set U, and G' represents a connected acyclic graph. Consequently, the problem of minimizing the pairwise distance is reduced to the well-known "minimum routing cost spanning tree"* problem. It is worth noting that the reduced problem has been proven to be NP-hard [35], [36]. As a result, our problem is also classified as NP-hard. ■

*. It is also called the optimum distance spanning tree, shortest total path length spanning tree, minimum total distance spanning tree, or minimum average distance spanning tree.

Algorithm 1 DSJA Routing Algorithm

Input:
All potential SFUs: $\mathbf{S}$;
All existing clients: $\mathbf{U}$;
Directed graph that contains existing clients and all SFUs: $G = (V, E)$;
Distance Matrix of all nodes: D;
New user: x;
Streaming paths between nodes: $P = p_{i,j}, \forall i \in \mathbf{V}, \forall j \in \mathbf{V}, p_{i,j} \subset E$;
Output:
New graph after user joins the video session: G_{new};
Paths from new user to existing users P;

for $i \in S$ **do**
  c_i = current average job queuing delay on the SFU i;
  Edge(V_{i_in}, V_{i_out}) = c_i;
end for
$Cost_s = 0$ where $\forall i \in \mathbf{S}$;
for $s \in S$ **do**
  $G = G + edge(x, s\_in) + edge(s\_out, x)$;
  for $u \in U$ **do**
    $D(x, u) = D(x, s\_in) + D(s\_in, u)$
    $D(u, x) = D(u, s\_out) + D(s\_out, x)$
    $Cost\_s+ = D(x, u) + D(u, x)$
  end for
  Delete $edge(x, s\_in)$ and $edge(s\_out, x)$ in G;
end for
Find the corresponding s with minimum $Cost\_s$
$G = G + edge(x, s\_in) + edge(s\_out, x)$;
for $v \in V$ **do**
  $p_{x,v} = edge(x, s\_in) \cup p_{s\_in,v}$;
  $p_{v,x} = p_{v,s\_out} \cup edge(s\_out, x)$
  $P = P + p_{x,v} + p_{v,x}$
end for
return G, P

5.3 Routing Problem Relaxation

This route scheduling problem is NP-hard and difficult to solve due to its algorithmic complexity. Finding an optimal solution requires enumerating all possible SFUs combinations for all users, resulting in a time complexity of $O(M^N)$, where N is the number of users and M is the number of SFUs. Such complexity is unacceptable, particularly in scenarios such as online education where the number of users in a video conference can be very large. Therefore, we need to find an alternative solution.

Our proposed solution is based on the observation that users are discouraged from switching SFUs during video conference sessions due to the easily perceived latency variation that can affect their quality of experience. When users experience severe lag or network conditions, we disconnect and then re-connect them to the video session to change their streaming route. In this way, we only need to consider scheduling the path for a new user, and we simplify the problem from finding the optimal SFUs combination for all users to a single-user route scheduling problem.

To formalize this relaxed problem, we define a graph G = (V, E) that represents the current SFUs configuration, and given a new user u, we seek to find the SFU $v \in V$ such

that the cost of adding u to v is minimized. This single-step problem can be efficiently solved using a shortest-path-based algorithm with acceptable algorithmic complexity. Once we have calculated the optimal routing path, we delete any unused edges between the user and SFUs to ensure that each user will not connect to other SFUs in future solutions to the problem.

The final algorithm we propose is shown in Algorithm 1. First, we update the cost (job queuing delay) of each SFU (Lines 1-3). Next, we enumerate all potential SFUs (Lines 6-7) and calculate the total cost (latency) of the video conference (Lines 8-13). After finding the best solution, we maintain the active edges (Lines 15-20).

6 Streaming Joint Adaptation

In this section, we firstly prove the NP-hardness in Section 6.1, followed by insights of designing streaming adaptation algorithm. We then introduce several relaxations in Section 6.2, converting the original problem to a one-step joint adaptation problem. This conversion turns the long-term dependent problem into a problem that can be solved separately at each period and allows us to resolve the joint adaptation challenge without any knowledge of future predictions. Lastly, we propose streaming adaptation algorithm in Section 6.3, which takes the coupling constraint and running speed into full consideration with superiority.

6.1 NP-hardness and Designing Insights

Theorem 2. The server-driven joint loss and bitrate adaptation problem towards maximized overall QoE is NP-hard.

Proof: As the original problem is complex, we consider its simplified case. We first assume that each sender has sufficient encoding ability, i.e., every sender can encode its video fast enough at all bitrate levels. Next, remove the upload capacity constraints so the server can forward the video from any sender at any bitrate level. In this way, we transfer the original problem to a simplified version, that is, assigning senders' video with optimal bitrate (i.e., items) to each receiver (i.e., knapsack) so that the overall QoE (i.e., the total value of the items in all knapsacks) is maximized while the total used bandwidth of each receiver (i.e., the sum of weight for each knapsack) does not exceed the downlink bandwidth capacity (i.e., B_j^{down}). Thus, we can reduce the multiple knapsack (MKP) problem, a known NP-hard problem, to our simplified problem. As our simplified problem is a special case of the original problem, the joint adaptation problem is NP-hard. ■

Although we can solve the NP-hard problem with a time-average objective over a finite horizon by Dynamic Programming (DP) [37] approaches, it is still difficult to achieve an efficient and effective solution using traditional DP-based methods for two main reasons. First, the original problem is much more complex with more constraints. The large state space under a DP formulation consists not only of the timeslot index k and network conditions but also of predicted encoding delay $T(\mathbf{r}_k^i)$, bringing significant challenges to achieving fast or even realtime decisions. Second, the original problem formulation aims to tackle the time-average joint adaptation problem, which needs predictions of future dynamics. In order to overcome the above challenges related to traditional DP-based approaches, we should take an alternative approach.

6.2 Problem Relaxation

Considering the original time-average joint adaptation problem formulation, we start by putting forward three relaxations. Instead of considering a limited time, we first explore the joint adaptation problem in the limiting regime when the video meeting becomes long, i.e., $K \to \infty$. Thus, the objective and the third constraint can be expressed as follows respectively:

$$\max_{\{r_k^{i,j}, c_k^i\}} \quad \lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} \left[\sum_{j=1}^{J} \eta_j \cdot QoE_j(k) \right] \tag{7}$$

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} d_{i,j}(k) \le \lambda T, \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \tag{8}$$

Next, we simplify the upload bandwidth constraint by eliminating its maximum operation:

$$\frac{r_k^{i,j}}{c_k^i} \le B_i^{up}(k), \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \tag{9}$$

We do the third relaxation by constructing rate stable virtual queues $Q_{i,j}(k)$ for each pair of sender and receiver as follows:

$$Q_{i,j}(k+1) = \max\{Q_{i,j}(k) + d_{i,j}(k) - \lambda T, 0\} \tag{10}$$

Lemma 1. If the virtual queue $Q_{i,j}(k)$ is rate stable, i.e., $\lim_{K\to\infty} \frac{Q_{i,j}(K)}{K} \le 0$, then we have $\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} d_{i,j}(k) \le \lambda T$.

Proof: According to (10), we can get:

$$\begin{aligned} & Q_{i,j}(k+1) - Q_{i,j}(k) \\ = \ & \max\{d_{i,j}(k) - \lambda T, -Q_{i,j}(k)\} \\ \ge \ & d_{i,j}(k) - \lambda T \end{aligned} \tag{11}$$

By taking the average from time index 0 to time index $K-1$ on both sides of (11) and letting K goes to infinite, we have

$$\lim_{K\to\infty} \frac{Q_{i,j}(K)}{K} \ge \lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} d_{i,j}(k) - \lambda T \tag{12}$$

Thus, if the virtual queue $Q_{i,j}(K)$ is rate stable, we have

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} d_{i,j}(k) \le \lambda T \tag{13}$$

This completes the proof. ■

By introducing the rate stable virtual queue $Q_{i,j}(k)$, the constraint (8) can be rewritten as:

$$\lim_{K\to\infty} \frac{Q_{i,j}(K)}{K} \le 0, \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \tag{14}$$

Taking advantage of the above relaxations and motivated by Lyapunov Optimization [38]–[40], we can further transfer the time-average problem to a one-step optimization problem so that the joint adaptation challenge can be resolved separately at each time slot. Before calculating the

Lyapunov penalty, we first introduce a concatenated vector of the virtual queues as:

$$\Theta_j(k) \triangleq [Q_{1,j}(k), Q_{2,j}(k), ..., Q_{I,j}(k)] \tag{15}$$

Thus, the corresponding Lyapunov penalty function for every receiver can be calculated as:

$$P(\Theta_j(k)) \triangleq \frac{1}{2}\sum_{i=1}^{I} Q_{i,j}(k)^2 \tag{16}$$

With (16), the Lyapunov penalty drift $\Delta(\Theta_j(k))$ is:

$$\Delta(\Theta_j(k)) = E\{P(\Theta_j(k+1)) - P(\Theta_j(k))|\Theta_j(k)\} \tag{17}$$

Enforcing the virtual queue constraints is equivalent to minimizing the drift penalty. Thus, we can turn the original maximization problem into a minimization problem so that maximizing the overall QoE with virtual queue constraint is equivalent to minimizing the following drift-plus-penalty term:

$$\sum_{j=1}^{J}\Big[\Delta(\Theta_j(k)) + V_j(-\eta_j QoE_j(k))\Big] \tag{18}$$

where V_j is a non-negative weight to allow a trade-off between the virtual queue constraint and the objective of overall QoE maximization.

Therefore, original problem formulation (6) can be transformed to the one-step problem as follows:

$$\begin{aligned}\min_{\{r_k^{i,j},c_k^i\}} \quad & \sum_{j=1}^{J}\Big[\Delta(\Theta_j(k)) - V_j(\eta_j QoE_j(k))\Big] \\ s.t. \quad & \tfrac{r_k^{i,j}}{c_k^i} \le B_i^{up}(k), \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \\ & \sum_{i=1}^{I}\tfrac{r_k^{i,j}}{c_k^i} \le B_j^{down}(k), \forall j \in \mathbf{J} \\ & r_k^{i,j} \in \mathbf{R}, c_k^i \in \mathbf{C}\end{aligned} \tag{19}$$

Furthermore, we can simplify the formulation of (19) by finding the upper bound of the penalty drift in (17).

$$\begin{aligned}& P(\Theta_j(k+1)) - P(\Theta_j(k+1)) \\ =& \frac{1}{2}\sum_{i=1}^{I}\Big[Q_{i,j}(k+1)^2 - Q_{i,j}(k)^2\Big] \\ \le& \frac{1}{2}\sum_{i=1}^{I}\Big[d_{i,j}(k) - \lambda T\Big]^2 + \sum_{i=1}^{I} Q_{i,j}(k)\Big[d_{i,j}(k) - \lambda T\Big]\end{aligned} \tag{20}$$

Substituting (20) to (17), we can get:

$$\Delta(\Theta_j(k)) \le C_j + \sum_{i=1}^{I} Q_{i,j}(k)\Big[d_{i,j}(k) - \lambda T\Big] \tag{21}$$

where C_j is constant that always satisfies the below condition:

$$C_j \ge \frac{1}{2}\sum_{i=1}^{I} E\Big[(d_{i,j}(k) - \lambda T)^2|\Theta_j(k)\Big] \tag{22}$$

Thus, after the further Lyapunov optimization and the upper bound substituting, the original time-average problem is now simplified to the following one-step joint adaptation problem:

$$\begin{aligned}\min_{\{r_k^{i,j},c_k^i\}} \quad & \sum_{j=1}^{J}\Big\{\sum_{i=1}^{I} Q_{i,j}(k)\Big[d_{i,j}(k) - \lambda T\Big] - V_j(\eta_j QoE_j(k))\Big\} \\ s.t. \quad & \tfrac{r_k^{i,j}}{c_k^i} \le B_i^{up}(k), \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \\ & \sum_{i=1}^{I}\tfrac{r_k^{i,j}}{c_k^i} \le B_j^{down}(k), \forall j \in \mathbf{J} \\ & r_k^{i,j} \in \mathbf{R}, c_k^i \in \mathbf{C}\end{aligned} \tag{23}$$

6.3 Streaming Joint Adaptation Algorithm

The one-step problem in Section 6.2 still need to make decisions on the two separated action space, and it still need to deal with the coupling objective. To tackle these difficulties, we propose our streaming joint adaptation algorithm based on primal decomposition, which can obtain the near-optimal solution quickly on the server. We first rephrase the formula in (23) as follows:

$$\begin{cases} f_0(\mathbf{r}_j, \mathbf{c}, k) = \sum_{i=1}^{I} Q_{i,j}(k)\Big[d_{i,j}(k) - \lambda T\Big] - V_j(\eta_j QoE_j(k)) \\ f_1(\mathbf{r}_j, \mathbf{c}, k) = \frac{r_k^{i,j}}{c_k^i} - B_i^{up}(k) \\ f_2(\mathbf{r}_j, \mathbf{c}, k) = \sum_{i=1}^{I}\frac{r_k^{i,j}}{c_k^i} - B_j^{down}(k) \end{cases} \tag{24}$$

where $\mathbf{r}_j = [r_k^{1,j}, r_k^{2,j}, ..., r_k^{I,j}]^T$ and $\mathbf{c} = [c_k^1, c_k^2, ..., c_k^I]^T$.

Thus, (23) can be represented in the following format:

$$\begin{aligned}\min_{\{\mathbf{r}_j\},\mathbf{c}_k} \quad & \sum_{j=1}^{J} f_0(\mathbf{r}_j, \mathbf{c}, k), \\ s.t. \quad & f_n(\mathbf{r}_j, \mathbf{c}, k) \le 0, n = 1, 2 \\ & r_k^{i,j} \in \mathbf{R}, c_k^i \in \mathbf{C}\end{aligned} \tag{25}$$

To cope with the coupling problem, as both bitrate and FEC code rate determine the delay metric, we transfer the original problem into one master problem and several subproblems based on the primal decomposition. We introduce a master agent to account for the information update and action aggregations as shown in (25). We then introduce J slave agents to be responsible for subproblem optimization within each iteration, and every slave agent simulates a receiver to solve the following subproblem with only linear constraints:

$$\mathbf{r}_j^* = inf\{f_0|f_n \le 0, n = 1, 2\} \tag{26}$$

We now design the algorithm for joint loss and bitrate adaptation as shown in Algorithm 2, which can be implemented on the server with slave agents running in parallel. The required inputs of state information at each time slot include network conditions such as bandwidth capacity and channels' loss rates, as well as receivers' bitrate requests. Also, the parameters consist of maximum iteration between master agent and slave agents at each time slot and the incremental extent of the objective, which are responsible for the stop criteria.

Algorithm 2 DSJA Streaming Adaptation Algorithm (for joint loss and bitrate)

Input:
Total times: K; Clients: $\mathbf{I}, \mathbf{J}$;
Bandwidth capacity: $B_i^{up}(k)$,$\forall i \in \mathbf{I}$; $B_j^{down}(k)$, $\forall j \in \mathbf{J}$;
Bitrate requests: $\hat{r}_k^{i,j}$, $\forall i \in \mathbf{I}, \forall j \in \mathbf{J}$;
Channel's loss rate: $p_i(k)$, $\forall i \in \mathbf{I}, \mathbf{J}$;
Threshold: ζ; Maximum iterations: n;
Output:
Optimal bitrate and FEC code rate: $r_k^{i,j}, c_k^i$;
Initialize $Q_{i,j}(k)$, $i = 1, 2, ...I$, $j = 1, 2, ..., J$;
for $k = 1$ to K **do**
  Initialize $iteration = 0$, $\triangle = \infty$;
  while $iteration < n$ and $\triangle > \zeta$ **do**
    for all $i, \in \mathbf{I}, j \in \mathbf{J}$ **do**
      $r_k^{i,j} = \underset{\mathbf{r}}{inf}\{f_0 | f_n \leq 0, n = 1, 2\}$;
      Update $\mathbf{r}_j$ to the master agent;
    end for
    for all $i \in \mathbf{I}, j \in \mathbf{J}$ **do**
      $r_k^{i,j} = \underset{r \in \mathbf{R}}{\arg\min} |f_0^j(r_k^{i,j}) - f_0^j(r)|$;
      $c_k^i = \underset{c \in \mathbf{C}}{\arg\min} \sum_{j=1}^{J} f_0^j(c)$;
    end for
    $\triangle = \sum_{j=1}^{J} f_0^j(k) - f_0^j(k-1)$;
    $iteration++$;
  end while
  Output joint adaptation solutions $\mathbf{r}$ and $\mathbf{c}$ for time slot k;
end for

In each iteration, every slave agent will first perform the subproblem optimization step in (26) on behalf of the corresponding receiver. As only one variable exists in the subproblem, every slave agent will roll a one-step optimization. They first adopt the fixed FEC code rate from the server agent to obtain the optimal bitrate choices for every receiver. It is worth noting that the encoding type constraints are omitted by every slave agent so that the original subproblem can be relaxed and transformed into a linear constraint problem, thus reducing the complexity of the solution. After that, every slave agent will update their optimal solutions to the master agent responsible for aggregations. Without bitrate clustering, It will assign the best encoding bitrate type according to the optimal bitrate solutions from slave agents through minimum objective sacrifice. Similarly, we can generate an optimal FEC code rate for each sender by minimizing the overall penalty drift.

The iteration will stop until it reaches the maximum iteration setting or the extent of the overall performance improvement is insignificant. Opposite to DP-based approaches, we emphasize that our algorithm has the superiority in processing time with time complexity of $O(nIJ)$ since it significantly increases the scalability by translating the original problem to a master problem and several one-variable subproblems with linear constraints. Thus, the optimal decisions to encode on senders and to stream forwarding on the server can be effectively and efficiently executed to achieve the optimized overall QoE.

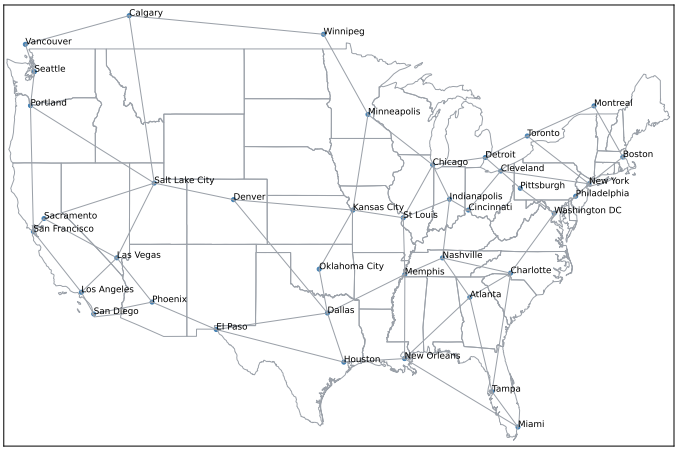


Fig. 6. The topology of simulated network.

7 Performance Evaluation

In this section, we compare the DSJA framework with state-of-the-art MRVS approaches and evaluate their performance with extensive trace-driven experiments.

7.1 Methodology

Network Topology: To simulate the real-world topology, we build the SFU nodes following the US backbone network and use famous universities as client nodes. Fig. 6 shows the structure of the backbone network, which is provided by the SNDlib project [41]. We set the capacity of each link in the network as 100Mbit/s and the delay increases by 1ms every 100Km.

Network Traces: To evaluate solutions under realistic network conditions, we create a corpus of network traces for more than 50 hours by combining two public datasets: (1) the FCC dataset [42] containing over 1 million throughput traces, each of which logs the average throughput over 2100 seconds. (2) the HSDPA dataset [43] on mobile throughput covers multiple usage scenarios such as buses, trains, and cities. To avoid trivial bitrate and FEC code rate selection, the synthesized corpus only considers the original traces with average throughput between 0.3 Mbps and 5 Mbps. Moreover, we collect the loss rate in two lossy network channels with an average of 1% and 2% loss rates.

Baselines: We consider the following methods as the baselines, which apply different SFU architectures (as shown in Fig. 7), for comparison with DSJA in MRVS scenarios:

- *MTL:* An adaptive bitrate control algorithm of *MultiLive* for the multi-party realtime scenarios. It is a server-driven approach and comprehensively considers the uplink bandwidth and downlink bandwidth of all clients. *Single-SFU* architecture is applied in the experiment.
- *OPG:* An integrated *Oppugno* framework that achieves joint loss and bitrate adaptation towards maximized QoE in realtime streaming services, which employs deep reinforcement learning algorithms on the receiver side. It is integrated with *Edge-SFUs* architecture in the experiment.
- *SS:* Adopts *SVC* for sender-side video encoding and *Single-SFU* architecture to select and forward streams with scalability.

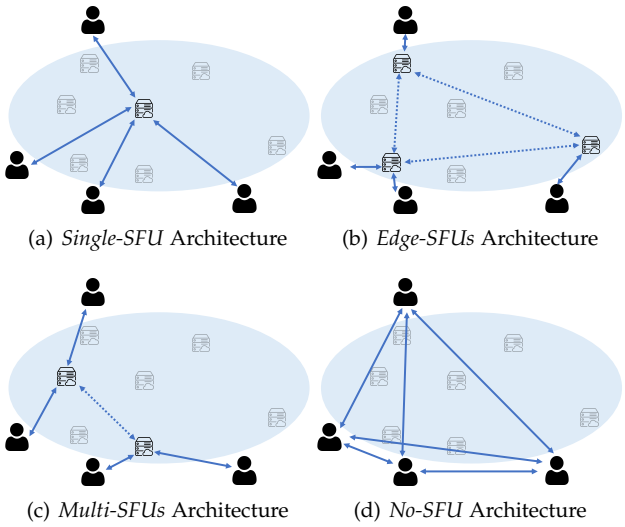


Fig. 7. Figure 7 illustrates four different SFU architectures used in our study. The *Single-SFU* Architecture (a) uses one SFU to perform selective forwarding for all clients in a meeting. The *Edge-SFUs* Architecture (b) allows each client to select the closest SFU, which performs the corresponding selective forwarding. The *Multi-SFUs* Architecture (c) uses dynamic server joint adaptation (DSJA) to select the optimal SFU based on network conditions and geographical distances. Finally, the *No-SFU* Architecture (d) has no SFU performing selective forwarding; instead, clients send data directly to each other.

- *MESH:* A simple version of *MTL* where *SVC* and *SFU* architectures are not applied.

Parameter Setting: Given the bandwidth traces, we set the available video bitrate as: $\{0.3, 0.5, 1.0, 2.0, 3.0, 5.0\}$ Mbps. We also set the available FEC code rates to range from 0.9 to 1.0. For the QoE weight setting, we choose $\{1, 1, 1, 2.5, 0.1\}$ and $\{1, 1, 1, 1.5, 0.25\}$ for perceptual quality, fluctuation, bitrate mismatch, loss damage, and streaming delay, respectively. Note that the loss damage weight 0.25 directly maps to the degradation of raw video quality after loss recovery, which is a non-negligible item. In that case, the two QoE preferences are delay-sensitive and loss-sensitive types. Besides, we set all V_j as 1 to allow the balance between the time-average delay requirement and the objective of overall QoE maximization. Lastly, we set the maximum iteration in the DSJA algorithm as 2 to ensure the fast response of the server-side orchestration.

Experiment Setup: To illustrate how effective our server-driven approach can achieve when facing multiple clients with different viewing preferences, we consider two cases with 5, 10, 15, 20, 25, and 30 clients in random USA locations. In the first case, every client in the meeting requests two videos from other clients simultaneously. More specifically, one video is in a loss-sensitive type, and the other is in a delay-sensitive type. Therefore, the client has different viewing preferences for different videos on its own screen. The detailed results are demonstrated in Section 7.2. In the second case, every client in the meeting requests videos from all others; That is, each client can communicate with anyone in the video conferencing service. Moreover, we set some clients in the conference as delay-sensitive users, meaning these clients will adopt delay-sensitive QoE models wherever the video is from. The remaining clients in the meeting are loss-sensitive users, and they will adopt loss-sensitive QoE models. The experiment results are revealed in Section 7.3.

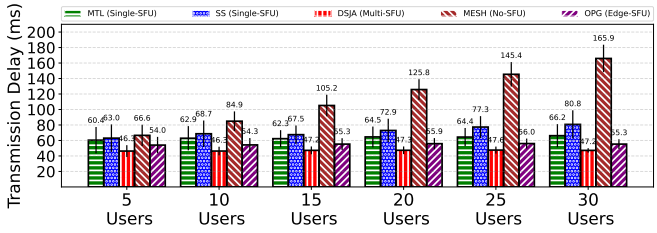


Fig. 8. The average Transmission Latency over synthesized dataset with real-world data

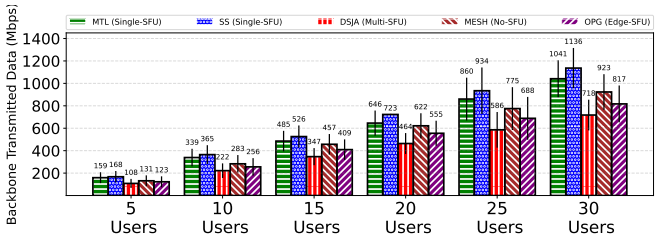


Fig. 9. The average Transmitted Data in the backbone network over a synthesized dataset with real-world data

7.2 Performance with Real-world Data

In this section, we compare the performance of the DSJA framework with other baselines. Based on our generated corpus, we evaluate the performance with two series of experiments with various loss conditions of 1% and 2% on average. Note that each viewer requests two videos with loss-sensitive and delay-sensitive QoE preferences. Also, only DSJA adopts *Multi-SFUs* architecture and chooses optimal SFUs for MRVS services. Other baseline methods use one central SFU, edge SFUs, or no SFU. The average transmission latency of DSJA and other baselines are shown in Fig. 8. The average amount of Transmitted Data per second in the backbone network is illustrated in Fig. 9, and the detailed CDF plots are in Fig. 10. The overall QoE achieved by DSJA and other baselines are shown in Fig. 11, and the detailed CDF plots are in Fig. 12. There are five key observations as follows.

Firstly, as seen in Fig. 8, when compared with all other baselines, DSJA significantly reduces the transmission latency, with 25.9% lower latency than *MultiLive* and 14.7% lower latency than *Oppugno*, proving the joint effectiveness of Route Scheduling and Stream Joint Adaptation algorithms. The *Multi-SFUs* architecture enables lower latency across a wide range of user locations distribution. One reason for this improvement is that DSJA enables more sender-receiver pairs to have shorter transmission distances. Also, DSJA considers the job queuing time of SFUs and network congestion levels of backbone links. In the first experiment case, when there are two senders, the increasing number of users has almost no effect on the transmission latency of DSJA and little influence on those of *SS*, *MTL* and *OPG*. The increasing latency of *MESH* is obvious.

Secondly, DSJA succeeds in releasing the backbone network pressure, as shown in Fig. 9. Using the DSJA algorithm, the average data transmitted per second in the backbone network is 31.1% lower than *MultiLive*, 14.0% lower

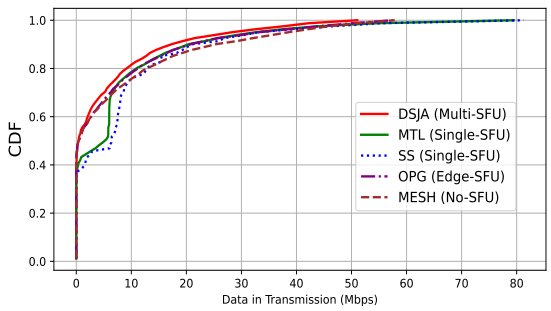
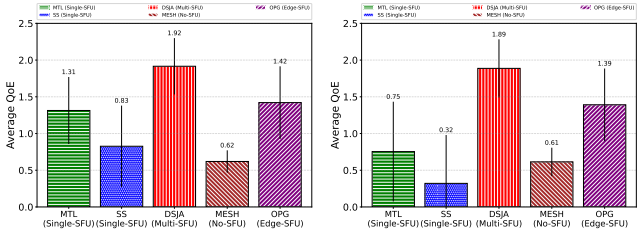


Fig. 10. CDF of average Data Transmitted in the Backbone Network over synthesized dataset with real-world data.



(a) The overall QoE over synthesized dataset with 1% loss (b) The overall QoE over synthesized dataset with 2% loss

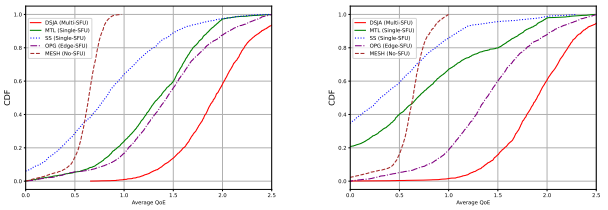
Fig. 11. The overall QoE over a synthesized dataset with real-world data.

than *Oppugno*, 36.6% lower than *SS*, and 22.6% lower than *MESH*. The main reason for this improvement is that under the *Multi-SFUs* architecture, the SFUs, where the selective forwarding is executed, are closer to the receivers. Another observation is that when there are only two senders, *Single-SFU* architecture generates the most data in transmission, even more than the *MESH* method.

Thirdly, DSJA is able to outperform all other baselines with higher average QoE, with 46.6% higher QoE than *MultiLive* in low loss rate conditions and 35.9% higher QoE than *Oppugno* in high loss rate conditions. This result confirms the effectiveness of DSJA by integrating the loss and bitrate adaptation for overall QoE maximization. An interesting observation is that both *SS* and *MESH* are inefficient in dealing with the multi-party joint adaptation problem, achieving worse performance than the other three frameworks as they lack server-driven bitrate orchestration and adaptive loss control mechanisms.

Fourthly, DSJA can achieve more stable and concentrated QoE scores than all other baselines. The CDF curves in Fig. 12(a) and Fig. 12(b) show a smaller QoE variance of DSJA than *MultiLive* and *Oppugno*. Moreover, the QoE scores of *Oppugno* are mainly concentrated within the range of 1.0 to 2.0, while *MultiLive* is even lower when the loss rate is relatively high. Instead, the QoE scores of DSJA are mainly concentrated within the range of 1.5 to 2.5 in different loss conditions, reflecting that DSJA can well mitigate the loss impact and can better handle the global streams orchestration with various loss and network throughputs, including the poor network condition with a large loss. The remaining two frameworks achieve quite concentrated but poor QoE performance, which infers that they fail to provide satisfactory QoE in MRVS services.

The fifth observation is that the loss adaptation is essential in multi-party realtime video streaming services. As shown in Fig. 11(a) and Fig. 11(b), there is a noticeable QoE drop for *MultiLive* as the loss rate increased, which indicates that *MultiLive* is quite sensitive to packet loss problems and insufficient to handle the large loss conditions even with server-side bitrate orchestration. Thus, this observation again confirms the superiority of our joint adaptation in MRVS services.



(a) CDF of QoE with 1% loss (b) CDF of QoE with 2% loss

Fig. 12. CDF of QoE metrics with real-world data.

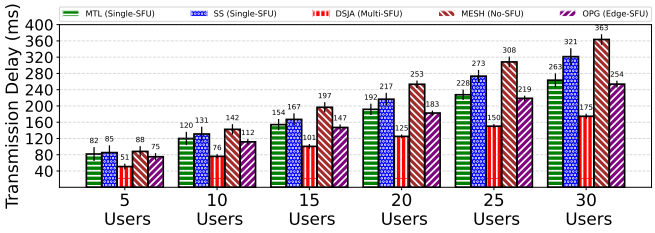


Fig. 13. The average Transmission Latency of a more generalized case. Each client requests videos from all others.

7.3 Performance of Generalization

In the experiments above, DSJA performs well on real-world data, while the realistic environment could be more complex with more viewing demands. To evaluate DSJA's ability to generalize to more demanding conditions, we conduct two more experiments with various loss rates and more video requests. To be more specific, we evaluate the DSJA framework and other baselines on the condition that every participant in the meeting room requests videos from all other participants. Moreover, among the participants, some serve as delay-sensitive users to attach great importance to streaming delay, while the other participants are loss-sensitive and adopt loss-sensitive QoE models. Transmission latency and the average amount of data in the backbone network transmission are illustrated in Fig. 13 and Fig. 14. The CDF details of the backbone network's transmitted data are shown in Fig. 15. We demonstrate the overall QoE achieved by DSJA compared with other baseline methods and the detailed CDF plots in Fig. 16. After the generalization experiments, we have another three critical observations.

Firstly, as the viewing demands of participants become more complex and generalized, DSJA still gets lower transmission latency and less data transmitted in the backbone network than all other baselines. And this improvement, brought by the *Multi-SFUs* MRVS architecture, becomes more obvious. As shown in Fig. 13, DSJA's transmission latency is 48.6% lower than *MESH*, 42.5% lower than *SS*, 31.6% lower than *Oppugno*, and 35.3% lower than *MultiLive*.

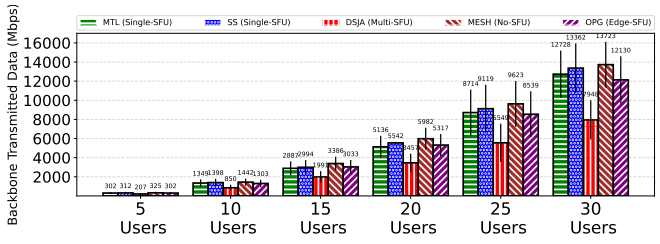


Fig. 14. The average Transmitted Data in the backbone network of a more generalized case. Each client requests videos from all others.

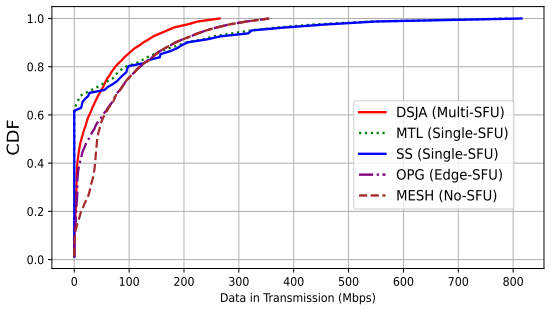
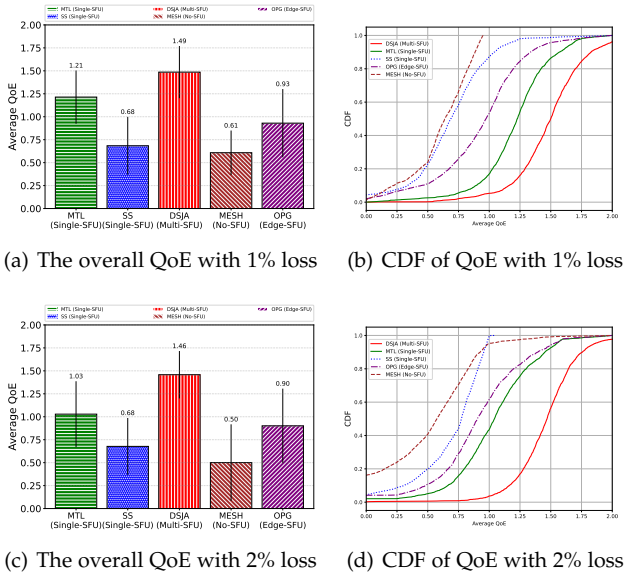


Fig. 15. CDF of average Data Transmitted in the Backbone Network of a more generalized case. Each client requests videos from all others.

Also, DSJA's average transmitted data per second in the backbone network is at least 34.2% lower than all other baseline methods, as seen in Fig. 14. One observation from Fig. 15 is that although under the *Single-SFU* structure, more links have low network pressure, there are also more links that need to transmit extremely large volumes of data. The main reason behind this phenomenon is that *Single-SFU* methods tend to have a much higher utilization rate of network links around the chosen single SFU, causing a low utilization rate of network links far from it. On the contrary, DSJA uses more SFUs and considers the congestion level of each network link. As a result, network congestion is more likely to incur under the *Single-SFU* structure. The total average amount of transmitted data per second in the backbone network is also higher when compared with the DSJA method that applies the *Multi-SFUs* architecture.

Secondly, the DSJA framework performs better than all other baseline methods with higher average overall QoE scores and more concentrated QoE distributions in the more generalized multi-party realtime video streaming scenarios. As shown in Fig. 16(a) and Fig. 16(c), the average overall QoE of the DSJA framework is at least 23.1% higher than other baseline methods in low loss rate conditions and is at least 41.7% higher than others in high loss rate conditions. Moreover, the QoE scores of DSJA are mainly concentrated within the range of 1.25 to 1.75, reflecting that DSJA can better mitigate the loss impact and solve the more challenging bandwidth allocation problem as the number of videos increases.

The third observation reveals the superiority of the server-driven joint adaptation design as the loss rate and the viewers' requested video number increase. Compared with the experiments in Section 7.2, the average overall QoE of all frameworks has been reduced. This result indicates that under the limited upload and download bandwidth, the video



(a) The overall QoE with 1% loss (b) CDF of QoE with 1% loss

(c) The overall QoE with 2% loss (d) CDF of QoE with 2% loss

Fig. 16. QoE performance of a more generalized case. Each client requests videos from all others.

quality of users will decrease with the increase of requested video numbers. However, server-driven approaches have better resistance to video quality degradation as the requested video number rises. Opposite to Fig. 11(a), Fig. 16(a) shows that the decline of QoE of DSJA and *MultiLive* are less than *Oppugno*, which obviously demonstrated the benefits of adopting server-driven strategies. Furthermore, in the high loss rate group shown in Fig. 16(c), the *MultiLive* solution with server-driven bitrate orchestration achieves similar QoE to the *Oppugno* solution with joint loss and bitrate adaptation, while the server-driven DSJA framework with joint loss and bitrate adaptation still performs the best. Thus, we can justify the importance of both the idea of server-driven and joint adaptation.

8 Related Work

Previous ABR algorithm mainly studies how to dynamically adjust the bitrate to utilize the bandwidth of the current network and improve the QoE. According to the decision basis, existing ABR algorithms are mainly divided into four categories: *1) rule-based ABR:* rule-based approaches mainly focus on the deviation of of certain indicator and adjust the bitrate. For example, Sun et al. [44] leveraged a data-driven approach to construct a Hidden-Markov-Model-based midstream predictor to model the stateful evolution of throughput. *2) hybrid ABR:* hybrid methods usually consider multiple factors when deciding the future bitrate. For example, Yin et al. [33] propose a model predictive control algorithm that can optimally combine throughput and buffer occupancy information to outperform traditional approaches. *3) model-based ABR [45]:* model-based approaches adjust the bitrate by formulating a more comprehensive model of QoE. For example, Zhang et al. [46] leverage the network dynamics feature of cellular networks and proposed an model predictive control (MPC)approach based on environment identification. *4) learning-based ABR [47]:* learning-based methods utilize machine learning algorithms to achieve bitrate adaptation. Zhang et al. [48] proposed a adaptive bitrate

selection algorithm based on meta-reinforcement learning to accommodate diverse network conditions and personalized QoE objectives. Mao et al. [2] pioneering trained a neural network model named Pensieve that selects bitrates for future video chunks based on observations collected by client video players. However, these studies on reliable TCP transmission and do not consider the packet loss problem, which can cause picture distortion and severely damage the users' QoE.

Some previous studies have put forward efforts on multi-party realtime video streaming services. Gao et al. [49] proposes an interest-aware rate adaptation approach that infers viewer interest based on video semantics and allocates bitrate budgets to improve QoE in video streaming, but there are limitations and challenges to implementing this approach in real-world scenarios. Lebreton et al. [50] addresses understanding and predicting user quitting ratio in adaptive bit rate video streaming through subjective experiments and proposes a framework with good prediction accuracy suitable for monitoring applications. Hajiesmaili et al. [18] cast a joint problem of user-to-agent assignment and transcoding-agent selection, and devised an adaptive parallel algorithm to simultaneously minimize the cost of the service provider and the conferencing delay. Hu et al. [21] considered the multi-server architecture and formulated server selection as a geometric problem to propose fast heuristics with theoretical worst-case guarantees. Wu et al. [17] presented a cloud-assisted mobile video conferencing system to improve the quality and scale of multi-party mobile video conferencing, where the decentralized algorithm can decide the best paths of streams and the most suitable surrogates for video transcoding along the paths to utilize the limited bandwidth fully. Moreover, Wang et al. [16] considered global optimization that takes into consideration of different QoE factors, and designed an adaptive bitrate control algorithm for the multi-party scenario by modeling the many-to-many ABR selection problem as a non-linear programming problem. These MRVS solutions mainly focus on the optimization problem with cost-efficiency, service scalability, and low latency. However, they lack the consideration of the overall orchestration with handling the potential loss problems, which may lead to severe damage to perceptual video quality [51].

9 Conclusion

In this paper, we propose DSJA, a distributed server-driven joint route scheduling and streaming adaptation framework in multi-party realtime video streaming services. Our goal is to maximize the overall quality of experience (QoE) for all users. We first emphasize the importance of route scheduling and server orchestration in MRVS, and abstracted its service model to better tackle the challenges of selecting the best SFUs, bitrate, FEC code rate, and streaming forwarding choices. Afterward, we investigated the QoE model and modelize the problems of maximizing overall QoE. We then proposed our two-step solutions including a route scheduling algorithm and a streaming adaptation algorithm. We evaluated the performance of our DSJA framework over a broad set of network conditions, real-world network topology, and different QoE metrics. The results indicate that our proposed framework outperformed existing advanced solutions by $23.1\% \sim 41.7\%$.

References

[1] K. Shen, D. Zhang, Z. Zhu, L. Zhang, F. Wang, and D. Wang, "Sja: Server-driven joint adaptation of loss and bitrate for multi-party realtime video streaming," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2023.

[2] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017, pp. 197–210.

[3] I. Sandvine, "Global internet phenomena report," *North America and Latin America*, 2023.

[4] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022," *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018.

[5] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017, pp. 183–196.

[6] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 495–511.

[7] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proceedings of the ACM International Conference on Multimedia*, 2018, pp. 1208–1216.

[8] L. Zhang, Y. Zhang, X. Wu, F. Wang, L. Cui, Z. Wang, and J. Liu, "Batch adaptative streaming for video analytics," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 2158–2167.

[9] M. Zhang, F. Wang, and J. Liu, "Casva: Configuration-adaptive streaming for live video analytics," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 2168–2177.

[10] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "Sensei: Aligning video streaming quality with dynamic user sensitivity," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021, pp. 303–320.

[11] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong, and L. Song, "Perm: Neural adaptive video streaming with multi-path transmission," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2020, pp. 1103–1112.

[12] X. Zuo, J. Yang, M. Wang, and Y. Cui, "Adaptive bitrate with user-level qoe preference for video streaming," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 1279–1288.

[13] L. Zhang, H. Guo, Y. Dong, F. Wang, L. Cui, and V. Leung, "Collaborative streaming and super resolution adaptation for mobile immersive videos," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2023.

[14] L. Sun, T. Zong, S. Wang, Y. Liu, and Y. Wang, "Towards optimal low-latency live video streaming," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2327–2338, 2021.

[15] D. Zhang, K. Shen, F. Wang, D. Wang, and J. Liu, "Towards joint loss and bitrate adaptation in realtime video streaming," in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, 2022, pp. 1–6.

[16] Z. Wang, Y. Cui, X. Hu, X. Wang, W. T. Ooi, Z. Cao, and Y. Li, "Multilive: Adaptive bitrate control for low-delay multi-party interactive live streaming," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 923–938, 2021.

[17] Y. Wu, C. Wu, B. Li, and F. C. Lau, "vskyconf: Cloud-assisted multi-party mobile video conferencing," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2013, pp. 33–38.

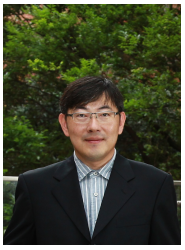
[18] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay design for multi-party cloud video conferencing," *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2760–2774, 2017.

[19] Zoom Corporation, "Meeting and phone statistics," https://support.zoom.us/hc/en-us/articles/202920719-Meeting-and-phone-statistics, 2021.
[20] J. Wang, G. Zhao, H. Xu, Y. Zhao, X. Yang, and H. Huang, "Trust: Real-time request updating with elastic resource provisioning in clouds," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 620–629.
[21] Y. Hu, D. Niu, and Z. Li, "A geometric approach to server selection for interactive video streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 840–851, 2016.
[22] Y. Yuan, W. Wang, Y. Wang, S. S. Adhatarao, B. Ren, K. Zheng, and X. Fu, "Vsim: Improving qoe fairness for video streaming in mobile environments," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 1309–1318.
[23] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018, pp. 44–58.
[24] Y. Chen, Q. Li, A. Zhang, L. Zou, Y. Jiang, Z. Xu, J. Li, and Z. Yuan, "Higher quality live streaming under lower uplink bandwidth: an approach of super-resolution based video coding," in *Proceedings of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2021, pp. 74–81.
[25] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, 2018.
[26] B. Wang, F. Ren, J. Yang, and C. Zhou, "Improving the performance of online bitrate adaptation with multi-step prediction over cellular networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 174–187, 2019.
[27] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2014, pp. 187–198.
[28] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.
[29] P. Dai, F. Song, K. Liu, Y. Dai, P. Zhou, and S. Guo, "Edge intelligence for adaptive multimedia streaming in heterogeneous internet of vehicles," *IEEE Transactions on Mobile Computing*, 2021.
[30] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 17, no. 9, pp. 1103–1120, 2007.
[31] T. Schierl, C. Hellge, S. Mirta, K. Gruneberg, and T. Wiegand, "Using h. 264/avc-based scalable video coding (svc) for real time streaming in wireless ip networks," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 3455–3458.
[32] J. Korhonen and P. Frossard, "Flexible forward error correction codes with application to partial media data recovery," *Signal Processing: Image Communication*, vol. 24, no. 3, pp. 229–242, 2009.
[33] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 325–338.
[34] F. Wang, C. Zhang, J. Liu, Y. Zhu, H. Pang, L. Sun *et al.*, "Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2019, pp. 910–918.
[35] D. S. Johnson, J. K. Lenstra, and A. R. Kan, "The complexity of the network design problem," *Networks*, vol. 8, no. 4, pp. 279–285, 1978.
[36] M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.
[37] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.
[38] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on average delays and queue size averages and variances in input-queued cell-based switches," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2001, pp. 1095–1103.
[39] N. Li, Y. Hu, Y. Chen, and B. Zeng, "Lyapunov optimized resource management for multiuser mobile video streaming," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 29, no. 6, pp. 1795–1805, 2018.
[40] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.
[41] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.
[42] Office of Engineering and Technology, "Raw data measuring broadband america 2021," https://www.fcc.gov/oet/mba/raw-data-releases, 2021.
[43] R. Haakon, V. Paul, G. Carsten, and H. Pål, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2013, pp. 114–118.
[44] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2016, pp. 272–285.
[45] C. Qiao, G. Li, Q. Ma, J. Wang, and Y. Liu, "Trace-driven optimization on bitrate adaptation for mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2243–2256, 2020.
[46] T. Zhang, F. Ren, W. Cheng, X. Luo, R. Shu, and X. Liu, "Towards influence of chunk size variation on video streaming in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 7, pp. 1715–1730, 2019.
[47] W. Li, J. Huang, J. Liu, W. Jiang, and J. Wang, "Learning audio and video bitrate selection strategies via explicit requirements," *IEEE Transactions on Mobile Computing*, 2023.
[48] W. Li, X. Li, Y. Xu, Y. Yang, and S. Lu, "Metaabr: A meta-learning approach on adaptative bitrate selection for video streaming," *IEEE Transactions on Mobile Computing*, 2023.
[49] G. Gao, H. Zhang, H. Hu, Y. Wen, J. Cai, C. Luo, and W. Zeng, "Optimizing quality of experience for adaptive bitrate streaming via viewer interest inference," *IEEE Transactions on Multimedia*, vol. 20, no. 12, pp. 3399–3413, 2018.
[50] P. Lebreton and K. Yamagishi, "Predicting user quitting ratio in adaptive bitrate video streaming," *IEEE Transactions on Multimedia*, vol. 23, pp. 4526–4540, 2020.
[51] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 1999, pp. 345–352.

Dayou Zhang (Student Member, IEEE) received the B.S. degree from the School of Information and Electronics, Beijing Institute of Technology, Beijing, China, in 2017. He is currently pursuing a Ph.D. degree in the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His areas of interest are multimedia networking, realtime video streaming, neural video coding, and machine learning.

Hao Zhu is currently an undergraduate student with the School of Data Science, The Chinese University of Hong Kong, Shenzhen. Starting in September 2023, he will begin pursuing his MSc degree with the Department of Computer Science, ETH Zurich, Switzerland. His research interests include multimedia networking and machine learning.

Kai Shen is currently pursing his M.A.Sc. degree at the Department of Electrical and Computer Engineering, University of Toronto, Canada. He received her B.Eng. degree in Computer Science and Engineering from The Chinese University of Hong Kong, Shenzhen. His research interests include computer networks, deep learning and federated learning.



Dan Wang (Committee Chair, IEEE) is currently a professor at the Department of Computing, The Hong Kong Polytechnic University. His research interests lie in networked systems, and recently in the inter-discipline domains of smart energy systems. He publishes in ACM SIGCOMM, ACM SIGMETRICS, IEEE INFOCOM, and in inter-discipline conferences, such as ACM e-Energy, ACM Buildsys. He won the Best Paper Award of ACM e-Energy 2018 and the Best Paper Award of ACM Buildsys 2018. He is currently the steering committee chair of IEEE/ACM IWQoS and the steering committee chair of ACM e-Energy. He is an advisor of EMSD, the Hong Kong SAR government. He has extensive experience in applying his research results to industry, including Huawei, IBM, Henderson, etc. He won the TechConnect Global Innovation Award in 2017.

Fangxin Wang (Member, IEEE) is an assistant professor at The Chinese University of Hong Kong, Shenzhen (CUHKSZ). He received the Ph.D., M.Eng., and B.Eng. degree all in Computer Science and Technology from Simon Fraser University, Tsinghua University, and Beijing University of Posts and Telecommunications, respectively. Before joining CUHKSZ, he was a postdoctoral fellow at the University of British Columbia. Dr. Wang's research interests include Multimedia Systems and Applications, Cloud and Edge Computing, Deep Learning and Big Data Analytics, Distributed Networking and System. He lead the intelligent networking and multimedia lab at CUHKSZ. He has published more than 30 papers at top journals and conferences papers, including INFOCOM, Multimedia, ToN, TMC, IOTJ, etc. He served as the publication chair of IEEE/ACM IWQoS, TPC member of IEEE ICC, and reviewer of many top conferences and journals, including INFOCOM, Multimedia, ToN, TMC, IOTJ.

Dear Editor and Reviewers,

I am writing to submit our paper titled “DSJA: Distributed Server-driven Joint Route Scheduling and Streaming Adaptation for Multi-Party Realtime Video Streaming” for publication consideration in IEEE Transactions on Mobile Computing (TMC).

As a preliminary version of this submission, a conference paper containing a subset of our results was presented published at IEEE INFOCOM 2023 [1]. In this current submission, **the most significant difference is that we extended the early work from a logical central forwarding unit (SFU) architecture to a practical distributed SFU-based forward architecture, and addressed the arising challenges therein**. We first add new challenges and our motivation for route scheduling and SFUs selection problem when considering real-world complex network topologies. We then developed a two-step modeling and inference framework for distributed multi-SFUs architecture in realtime video streaming. Our work significantly contributes to the existing research by providing a more comprehensive introduction, a new modeling and analysis of multi-SFUs selection and routing problems, a proof of NP-hardness of the routing problem, a greedy-based problem relaxation, a heuristic method algorithm with detailed illustration, and new comprehensive experimental results.

We would like to highlight the following key contributions of our TMC submission compared to the preliminary IEEE INFOCOM paper:

1. In Section 1, we extend the introduction and conduct a more comprehensive and clear description of problem definition and our contribution. Unlike previous research that assumed a logical centralized server that connects to all clients, we consider a much practical architecture that has distributed multiple SFUs. We add the new challenge under complex real-world topology network, which is a significant factor we need to consider in non-regional video streaming.
2. In section 2, we update our motivation that most previous work has neglected: no matter where the single SFU is located, clients have to send their video data through the SFU to reach other clients. This round-trip data transmission will significantly increase video latency. This dilemma provides much stronger motivation for our work and points out the potential improvement when applying multi-SFUs architecture.
3. The architecture in the original version only had a logical single central server. In the journal version, we proposed our multi-SFUs architecture in Section 3 to tackle the video latency issue. We introduced two logical components responsible for routing scheduling and streaming parameters adaptation in each SFU. This more comprehensive architecture can support clients that are far spread, or even on different continents.
4. In the conference version, the problem formulation in Section 4 only considered the adjustment of different streaming parameters. In the journal version, the architecture becomes much more complex. We define the Quality of Experience (QoE) maximization problem into a two-step adaptation process, and model these two problems into a route scheduling problem and a streaming parameter adaptation problem, respectively.
5. We added a new Section 5 to illustrate our efforts in tackling the emerging routing problem associated with multi-SFUs. We started with transferring the routing problem into a typical graph theory problem, and analyzed its NP-hardness. We proposed a relaxation that transformed the global route scheduling problem into a single-step shortest-path-like problem and developed a algorithm that could maintain the routing graph and find the near-optimal routing solution in lightweight calculations.
6. In the journal version of Section 7, we conducted more comprehensive experiments with significant improvement, compared with the conference version and other existing works. We applied the US backbone network topology provided by the SNDlib project to simulate the superiority of multi-SFUs architecture. We also extended our experiment with bandwidth usage to show the effectiveness of our method in saving bandwidth.

Overall, we believe our work well matches the field of mobile computing and non-regional video streaming and would be of interest to the readers of TMC.

Thank you very much in advance for handling this submission, and we look forward to your correspondence about its progress.

Sincerely,

Dayou Zhang, Hao Zhu, Kai Shen, Dan Wang, Fangxin Wang

[1] Kai Shen, Dayou Zhang, Zi Zhu, Lei Zhang, Fangxin Wang, Dan Wang, “SJA: Server-driven Joint Adaptation of Loss and Bitrate for Multi-Party Realtime Video Streaming”, in Proceedings of 2023 IEEE IEEE Conference on Computer Communications (INFOCOM), 2023.

SJA: Server-driven Joint Adaptation of Loss and Bitrate for Multi-Party Realtime Video Streaming

Kai Shen[1,2], Dayou Zhang[1,2], Zi Zhu[2], Lei Zhang[5], Fangxin Wang[2,1,3,4,*], Dan Wang[6]
[1]The Future Network of Intelligence Institute, The Chinese University of Hong Kong, Shenzhen
[2] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen
[3] Peng Cheng Laboratory
[4] The Guangdong Provincial Key Laboratory of Future Networks of Intelligence
[5] College of Computer Science and Software Engineering, Shenzhen University
[6]Department of Computing, The Hong Kong Polytechnic University
{kaishen, dayouzhang, zizhu}@link.cuhk.edu.cn, leizhang@szu.edu.cn
wangfangxin@cuhk.edu.cn, csdwang@comp.polyu.edu.hk

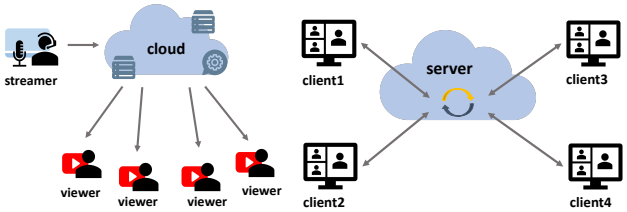
***Abstract*—The outbreak of COVID-19 has dramatically promoted the explosive proliferation of multi-party realtime video streaming (MRVS) services, represented by Zoom and Microsoft Teams. Different from Video-on-Demand (VoD) or live streaming, MRVS enables all-to-all realtime video communication, bringing significant challenges to service providing. First, unreliable network transmission can cause network loss, resulting in delay increase and visual quality degradation. Second, the transformation from two-party to multi-party communication makes resource scheduling much more difficult. Moreover, optimizing the overall QoE requires a global coordination, which is quite challenging given the various impact factors such as bitrate and loss.**

In this paper, we propose the SJA framework, which is, to our best knowledge, the first server-driven joint loss and bitrate adaptation framework in multi-party realtime video streaming services towards maximized QoE. We comprehensively design an appropriate QoE model for MRVS services to capture the interplay among perceptual quality, variations, bitrate mismatch, loss damage, and streaming delay. We mathematically formulate the QoE maximization problem in MRVS services. A Lyapunov-based relaxation and the SJA algorithm are further designed to address the optimization problem with close-to-optimal performance. Evaluations show that our framework can outperform the SOTA solutions by $18.4\% \sim 46.5\%$.

***Index Terms*—Multi-party realtime video streaming, Adaptive bitrate control, Forward error correction, Quality of Experience**

* Fangxin Wang is the corresponding author.

The work was supported in part by the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, by National Natural Science Foundation of China with Grant No.62102342, by Shenzhen Outstanding Talents Training Fund 202002, by Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and by The Major Key Project of PCL Department of Broadband Communication. Lei's work was supported in part by National Natural Science Foundation of China with Grant No.61902257 and No.62272317, by Guangdong Basic and Applied Basic Research Foundation under Grant No.2021A1515012633, by Shenzhen Science and Technology Program under Grant No.JCYJ20220531103402006. Dan's work was supported in part by RGC GRF 15210119, 15209220, 15200321, 15201322, CRF C5018-20G and from ITC via project "Smart Railway Technology and Applications" (No. K-BBY1), all from Hong Kong.



(a) Live Streaming (b) Video Conferencing

Fig. 1. The representative applications of realtime streaming services.

I. Introduction

The outbreak of COVID-19 has dramatically promoted the explosive proliferation of multi-party realtime video streaming (MRVS) services, which have profoundly affected how people live and communicate. Such applications represented by Zoom and Microsoft Teams are playing an indispensable role in various fields like online education, telemedicine, video conferencing, etc. Different from Video-on-Demand (VoD) services [1] represented by YouTube and crowdsourced live streaming services [2] shown in Fig. 1(a) represented by Twitch, multi-party realtime video streaming or conferencing shown in Fig. 1(b) embraces a new form of communication where each client is able to send its own stream to all other parties and simultaneously receive streams from others in real time. In the near foreseeable future, MRVS will continue to create tremendous revenue. According to the report from Cisco [3], realtime video streaming traffic has tripled within the last five years and reached 17% of Internet video traffic by 2022.

Providing satisfactory quality of experience (QoE) for users is always the key issue in MRVS, while new challenges arise in such a particular service. *The first one lies in the packet loss problem caused by the unreliable transmission in MRVS, which can further lead to delay increase and visual quality degradation if without careful processing.* Unlike TCP-based reliable transmission (e.g., VoD and live streaming), MRVS usually employs light-weight yet unreliable transmission protocols

based on UDP (or its variants together with some control protocols [4]) to keep fast responses. However, once packet loss occurs, current systems either wait for the re-transmission with higher delay, or directly deliver the incomplete stream to upper applications with potential video quality distortion.

The second challenge arises from the difficulty in coordinating the limited network resources given the transformation from two-party communication to multi-party communication. Traditional VoD or live streaming usually adopts the client-server architecture, where each client only needs to deal with the communication with the other side. MRVS is much more complex, where a central server node serves as the selective forwarding unit (SFU), responsible for receiving streams from different clients, orchestrating streams based on requests, and sending streams as a group to each destination. Previous works on bitrate adaptation [5], [6] can only handle the two-party communication scenario, while for the multi-party case, there is still a considerable gap in fine-grained resource allocation towards higher user QoE.

The third challenge is how to coordinate the transmission behaviors of different parties so as to maximize the overall QoE of a communication group. Conventional works [7]–[9] mostly start from a client-driven perspective and only make local decisions. Such an independent decision-making strategy can easily cause poor QoE from a global view. However, it is not easy to coordinate the interests of multiple parties, especially when considering a variety of impact factors such as loss, bitrate, and other network conditions.

Pioneer research works have made efforts toward these challenges. Sun et al. [10] developed Deep Reinforcement Learning frameworks to ensure low end-to-end video latency in live streaming services, but lack consideration for packet loss problems. Zhang et al. [11] proposed a joint bitrate and loss adaptation scheme for realtime video streaming, while they only focused on two-party communication. MultiLive [12] addressed the bitrate allocation in a multi-party live streaming scenario, but it only considered reliable transmissions. Therefore, existing works only partially tackle the challenges therein, calling for an integrated solution to address these crucial problems in delay, multi-stream orchestration, and multi-party coordination towards QoE maximization.

In this paper, we propose the SJA framework, a server-driven joint loss and bitrate adaptation framework in multi-party realtime video streaming services towards maximized QoE. In our architecture, senders first generate the initial streaming configurations based on their own policy (e.g., the Adaptive Bitrate (ABR) strategy). The central server then conducts a joint loss and bitrate adaptation decision with global orchestration based on the network conditions and viewer requests. Following the server-side decision, the senders will conduct the globally optimal stream configuration with scalable video coding (SVC), and the server will forward the optimal stream layer to the corresponding receivers. In summary, our contributions are as follows:

1) We investigate the problem caused by network loss in realtime video streaming and highlight the importance of joint adaptation of loss and bitrate in MRVS services. We further demonstrate the necessity of a server-driven architecture to optimize overall QoE.
2) To our best knowledge, our proposed framework is the first to consider a server-driven joint loss and bitrate adaptation and multi-party stream coordination in MRVS services. We comprehensively design an appropriate QoE model for MRVS services to capture the interplay among perceptual quality, variations, bitrate mismatch, loss damage, and streaming delay.
3) We formulate the QoE maximization problem in MRVS services, followed by relaxation and the SJA algorithm.
4) We conduct extensive realistic trace-driven evaluations. The results demonstrate that the SJA algorithm can achieve close-to-optimal performance with superior latency and loss alleviation.

TABLE I
SERVICE TYPES OF PIONEER RESEARCH

	realtime	multi-party	loss adaption	global coordination
DRL-Live [10]	✓			
Oppugno [11]	✓		✓	
vSkyConf [13]		✓		✓
AgRank [14]	✓	✓		
MultiLive [12]	✓	✓		✓
SJA	✓	✓	✓	✓

The remainder of the paper is organized as follows. Section II introduces the motivation of our framework. Section III illustrates the framework architecture. Section IV formulates the problem with the QoE model definition. Section V describes the design of the SJA algorithm in detail. We present the simulation results through extensive trace-driven experiments in Section VI. Section VII discusses the related work. Finally, we conclude the paper in Section VIII.

II. MOTIVATION

Researchers have become increasingly interested in multi-party realtime video streaming services given their great market value. Different from VoD or live services, MRVS has several key features: *1) realtime*: an acceptable transmission delay is usually below 150 ms [15], far less than VoD and live services; *2) simultaneous transceiving*: each node can send messages while at the same time receive messages from others, e.g., in a video conference; *3) multi-party*: multiple nodes can have all-to-all communication; *4) unreliable transmission*: existing systems mostly employ UDP-based protocol, which cannot guarantee the reliability in order to satisfy fast response.

The confluence of these features makes the MRVS service provision even more challenging towards maximized user QoE, calling for an integrated framework that jointly considers realtime, multi-party, loss adaptation, and global coordination for transmission optimization. Pioneer works [10]–[14] have

(a) 480p video with 1% loss (b) 480p video with 2% loss

(c) 1080p video with 1% loss (d) 1080p video with 2% loss

Fig. 2. Visual effects when multiple viewers experience different loss rates.

made attempts to address this problem. However, as summarized in Table I, they only consider partial factors and fail to achieve a comprehensive solution. Some other related works try to minimize the global network cost [16], reduce the end-to-end delay [17], or meet the fairness constraints [18], which also only tackle partial problems.

We have closely examined the features of MRVS services and summarized two insights as follows. *Firstly, we argue that extra bandwidth can be leveraged for redundant coding such that loss can be corrected without retransmission.* As shown in Fig. 2, even 1% of bit errors in the transmission can gravely damage viewers' perceptions. Even worse, in multi-party conferencing, if bit errors occur during video uploading, the user experience of its all viewers at different bitrate levels will be degraded. As a result, the impact of network loss problems induced by fast yet unreliable UDP should not be underestimated. Therefore, Forward Error Correction (FEC) mechanisms can be utilized to mask the effect of loss, where the extra bandwidth can be leveraged to facilitate loss recovery.

Secondly, we emphasize that the overall QoE maximization of MRVS can only be achieved through server-driven approaches. Studies towards QoE maximization usually focus on bitrate adaption, where an ABR controller will be responsible for selecting the most appropriate bitrate based on the available network throughput [8], [19], [20], receiver buffer occupancy [21], [22], or utilize reinforcement learning approaches [7]. These client-driven approaches only utilize local state information to request their own most suitable bitrates, while the decisions are usually not globally optimal in such a non-cooperative way. For example, if multiple viewers request the same video with different bitrates from a sender, it will increase the sender encoding time, resulting in higher latency for all receivers and failing to achieve satisfactory overall QoE.

Thus, motivated by non-negligible loss damages and the necessity of server-driven approaches, we take it one step further and explore, for the first time, the effects of server-driven joint loss and bitrate adaptation in MRVS.

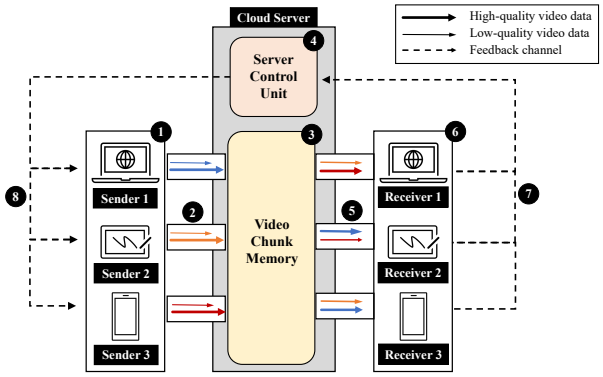


Fig. 3. SJA Framework

III. Framework Overview

The SJA framework for joint bitrate and loss adaptation in MRVS services towards overall QoE maximization. As illustrated in Fig. 3, SJA follows a server-driven architecture responsible for coordinating the streaming configuration of all clients. It mainly consists of four components, i.e., senders, receivers, video chunk memory and server control unit. It is worth noting that different from the logical distinction, a client in MRVS service can serve as both sender and receiver at the same time. The detailed functionalities are as follows:

- **Sender:** A sender is responsible for sending out streams with different bitrates employing scalable video coding [23]. SVC is an extension to the H.264/AVC [24] video codec standard, which can encode a video stream into a base layer and several enhancement layers. Besides, the sender also leverages redundant coding technique (i.e., FEC [25]) in our design, aiming to provide error recovery ability to avoid retransmission.
- **Receiver:** Since the receiver has all the local updated information, such as buffer occupancy and throughput estimation, it first conducts an initial request for bitrate using ABR algorithms, which have been widely explored by pioneer works [1], [9]. In order to support the global coordination, the feedback information such as the receiver's QoE, bitrate decision and network conditions needs to be sent back to the server for further control.
- **Server Control Unit:** The server control unit is the core component in the cloud server responsible for global coordination. It collects the state information from senders and receivers, as well as the receiver feedback information. Through our designed SJA algorithm (§V), it derives the best bitrate and loss adaptation decisions from a global view, and delivers the control signals to the senders for the next chunk streaming.
- **Video Chunk Memory:** Similar to a common selective forwarding unit (SFU) architecture, the video chunk memory at the cloud server will collect streams from multiple senders, conduct stream orchestration, and then

forward corresponding streams with appropriate bitrate to different receivers based on the decision result. Thanks to the SVC format, the video chunk memory is able to conduct light-weight layered streaming without much transcoding overhead.

Our framework operates every time cycle, which is a short time slot. The workflow of a decision cycle is demonstrated in Fig. 3. Every sender (❶) first generates an uplink video stream (❷) with multiple bitrate layers to the video chunk memory (❸) in the cloud server. After receiving the control signals from the server control unit (❹), the downlink streams (❺) with different layered bitrate and different coding redundancy will be forwarded to corresponding receivers (❻).

IV. QoE Model and Problem Formulation

We mathematically model the problem in this section to better understand the challenges under MRVS scenarios. To do so, we introduce our QoE design, which is the ultimate objective, followed by constructing the primary problem formulation and explaining the network constraints.

We assume that there are I senders and J receivers in the multi-party realtime video streaming service, where $\mathbf{I} = \{1, 2, ..., I\}$ represents the set of senders and $\mathbf{J} = \{1, 2, ..., J\}$ represents the set of receivers. Note that I may not equal J. Also, there are $\mathbf{R} = \{R_1, R_2, ..., R_q\}$ types of encoding bitrate levels and $\mathbf{C} = \{C_1, C_2, ..., C_p\}$ types of FEC code rate levels. Additionally, there are K timeslots, where $\mathbf{K} = \{0, 1, 2, ..., K\}$ represents the set of slotted time indexes.

A. QoE Model

Followed by existing works [2], [26], we consider five QoE metrics, i.e., *video rate based perceptual quality*, *perceptual quality variations*, *bitrate mismatch level*, *streaming delay* and *loss penalty*, where we separate the QoE degradation caused by bit errors from the original perceptual quality related to bitrate. Firstly, we can calculate the perceptual quality of sender i's video in receiver j at time slot k as follows:

$$q(r_k^{i,j}) = \log(\frac{r_k^{i,j}}{r_{min}}) \tag{1}$$

where $r_k^{i,j}$ denotes the bitrate level from sender i to receiver j at slotted time k, and r_{min} denotes the minimal bitrate level. Here we adopt the logarithmic function [22] to represent the video quality, where the ratio of real bitrate to minimal bitrate can be utilized to decrease the marginal quality improvement.

Secondly, we take advantage of the perceptual quality variations to penalize changes in video quality to favor smoothness [1]. In more detail, we adopt the absolute value of the difference between current video quality and the video quality of the last moment to represent the variations as follows:

$$\begin{aligned} |q(r_k^{i,j}) - q(r_{k-1}^{i,j})| &= |\log(\frac{r_k^{i,j}}{r_{min}}) - \log(\frac{r_{k-1}^{i,j}}{r_{min}})| \\ &= |\log(\frac{r_k^{i,j}}{r_{k-1}^{i,j}})| \end{aligned} \tag{2}$$

TABLE II
Notations used in this paper

Variables	Meaning
$r_k^{i,j}$	real bitrate from sender i to receiver j at time k
r_{min}	minimal bitrate level
$\hat{r}_k^{i,j}$	requested bitrate of sender i from receiver j at time k
$d_{i,j}(k)$	streaming delay between sender i and receiver j
c_k^i	FEC code rate of sender i at time k
$\mathbf{r}_k^i$	the set of all bitrates encoded by sender i at time k
$T(\mathbf{r}_k^i)$	encoding time of sender i at time k
$B_i^{up}(k)$	uplink bandwidth of sender i at time k
$B_j^{down}(k)$	downlink bandwidth of receiver j at time k
$p_i(k)$	loss rate between client i and the server at time k
$e_{i,j}(k)$	remaining loss rate from sender i to receiver j at time k
T	duration of each time slot

Thirdly, we argue that the assigned streaming may differ from the requested streaming of the receiver's ABR controller in practice. Thus, the bitrate mismatch penalty demonstrates the dissatisfaction between the receivers' requested bitrate and the transmitted bitrate. We can calculate it as follows:

$$B(r_k^{i,j}, \hat{r}_k^{i,j}) = \log(\frac{\hat{r}_k^{i,j}}{r_{min}}) - \log(\frac{r_k^{i,j}}{r_{min}}) = \log(\frac{\hat{r}_k^{i,j}}{r_k^{i,j}}) \tag{3}$$

The fourth metric denotes streaming delay as follows:

$$d_{i,j}(k) = T(\mathbf{r}_k^i) + \frac{\max_j\{r_k^{i,j}\}}{c_k^i \cdot B_i^{up}(k)} + \frac{\sum_{i=1}^{I} \frac{r_k^{i,j}}{c_k^i}}{B_j^{down}(k)} \tag{4}$$

whose three components are encoding latency, uploading latency, and downloading latency. Since we adopt one server model in this paper, we ignore the transmission time among different servers, the decision-making time, and the feedback transmission time. In the expression, $\mathbf{r}_k^i = \{r_k^{i,j} | r_k^{i,j} \in \mathbf{R}, j \in \mathbf{J}\}$ represents the set of all bitrates that sender i will encode at time k. Note that c_k^i is the FEC code rate, equal to k/n, where an encoder takes a block of k source symbols as inputs and generates a total of n FEC symbols ($n > k$) as output.

Lastly, we consider the QoE degradation caused by bit errors. We denote $e_{i,j}(k) = \max\{p_i(k) + p_j(k) - (1 - c_k^i), 0\}$ to represent the remaining bit error rate after the FEC recovery, and we map the bit errors to the QoE degradation penalty through $L(e_{i,j}(k))$, which will be analyzed in later section.

Therefore, we have considered all five QoE metrics in multi-party realtime video streaming. In conclusion, the QoE for receiver j at slotted time k is defined as:

$$\begin{aligned} QoE_j(k) = \sum_{i=1}^{I} \alpha_j^i \big\{ & \beta_j q(r_k^{i,j}) - \gamma_j |q(r_k^{i,j}) - q(r_{k-1}^{i,j})| \\ & - \delta_j B(r_k^{i,j}, \hat{r}_k^{i,j}) - \epsilon_j d_{i,j}(k) - \zeta_j L(e_{i,j}(k)) \big\} \end{aligned} \tag{5}$$

where α_j^i is the importance factor of receiver j towards the video from sender i, and other Greek letters represent the receiver's personalized viewing demand.

B. Problem Formulation

Integrating the viewers' personalized QoE demands and overall viewing architecture, we have the following optimization objective that aims to maximize the overall QoE driven by the server end under multi-party realtime video streaming:

$$\begin{aligned} \max_{\{r_k^{i,j}, c_k^i\}} \quad & \frac{1}{K}\sum_{k=0}^{K-1}\left[\sum_{j=1}^{J}\eta_j \cdot QoE_j(k)\right] \\ s.t. \quad & \max_j\{\tfrac{r_k^{i,j}}{c_k^i}\} \leq B_i^{up}(k), \forall i \in \mathbf{I} \\ & \sum_{i=1}^{I}\tfrac{r_k^{i,j}}{c_k^i} \leq B_j^{down}(k), \forall j \in \mathbf{J} \\ & \tfrac{1}{K}\sum_{k=0}^{K-1} d_{i,j}(k) \leq \lambda T, \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \\ & r_k^{i,j} \in \mathbf{R}, c_k^i \in \mathbf{C} \end{aligned} \tag{6}$$

where T is the duration of each time slot, and η_j represents the importance of receiver j in the meeting room. Our action space consists of the actual bitrate of each sender-receiver pair and the FEC code rate of every sender. The first constraint is the upload bandwidth constraint for each sender, and the second is the download bandwidth constraint for every receiver. The third constraint is the time-average expected delay requirement, which ensures the low latency of the service. The last one guarantees the available bitrate and FEC code rates, which belongs to pre-setted combinations and are non-negative.

V. SJA Algorithm for Joint Adaptation

In this section, we firstly prove the NP-hardness in Section V-A, followed by insights of designing SJA algorithm. We then introduce several relaxations in Section V-B, converting the original problem to a one-step joint adaptation problem. This conversion turns the long-term dependent problem into a problem that can be solved separately at each period and allows us to resolve the joint adaptation challenge without any knowledge of future predictions. Lastly, we propose SJA algorithm in Section V-C, which takes the coupling constraint and running speed into full consideration with superiority.

A. NP-hardness and Designing Insights

***Theorem** 1:* The server-driven joint loss and bitrate adaptation problem towards maximized overall QoE is NP-hard.

Proof: As the original problem is complex, we consider its simplified case. We first assume that each sender has sufficient encoding ability, i.e., every sender can encode its video fast enough at all bitrate levels. Next, remove the upload capacity constraints so the server can forward the video from any sender at any bitrate level. In this way, we transfer the original problem to a simplified version, that is, assigning senders' video with optimal bitrate (i.e., items) to each receiver (i.e., knapsack) so that the overall QoE (i.e., the total value of the items in all knapsacks) is maximized while the total used bandwidth of each receiver (i.e., the sum of weight for each knapsack) does not exceed the downlink bandwidth capacity (i.e., B_j^{down}). Thus, we can reduce the multiple knapsack (MKP) problem, a known NP-hard problem, to our simplified problem. As our simplified problem is a special case of the original problem, the joint adaptation problem is NP-hard. ■

Although we can solve the NP-hard problem with a time-average objective over a finite horizon by Dynamic Programming (DP) [27] approaches, it is still difficult to achieve an efficient and effective solution using traditional DP-based methods for two main reasons. First, the original problem is much more complex with more constraints. The large state space under a DP formulation consists not only of the timeslot index k and network conditions but also of predicted encoding delay $T(\mathbf{r}_k^i)$, bringing significant challenges to achieving fast or even realtime decisions. Second, the original problem formulation aims to tackle the time-average joint adaptation problem, which needs predictions of future dynamics. In order to overcome the above challenges related to traditional DP-based approaches, we should take an alternative approach.

B. Problem Relaxation

Considering the original time-average joint adaptation problem formulation, we start by putting forward three relaxations. Instead of considering a limited time, we first explore the joint adaption problem in the limiting regime when the video meeting becomes long, i.e., $K \to \infty$. Thus, the objective and the third constraint can be expressed as follows respectively:

$$\max_{\{r_k^{i,j}, c_k^i\}} \quad \lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}\left[\sum_{j=1}^{J}\eta_j \cdot QoE_j(k)\right] \tag{7}$$

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} d_{i,j}(k) \leq \lambda T, \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \tag{8}$$

Next, we simplify the upload bandwidth constraint by eliminating its maximum operation:

$$\frac{r_k^{i,j}}{c_k^i} \leq B_i^{up}(k), \forall i \in \mathbf{I}, \forall j \in \mathbf{J} \tag{9}$$

We do the third relaxation by constructing rate stable virtual queues $Q_{i,j}(k)$ for each pair of sender and receiver as follows:

$$Q_{i,j}(k+1) = \max\{Q_{i,j}(k) + d_{i,j}(k) - \lambda T, 0\} \tag{10}$$

Lemma 1: If the virtual queue $Q_{i,j}(k)$ is rate stable, i.e., $\lim_{K\to\infty}\frac{Q_{i,j}(K)}{K} \leq 0$, then we have $\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} d_{i,j}(k) \leq \lambda T$.

Proof: According to (10), we can get:

$$\begin{aligned} & Q_{i,j}(k+1) - Q_{i,j}(k) \\ = \ & \max\{d_{i,j}(k) - \lambda T, -Q_{i,j}(k)\} \\ \geq \ & d_{i,j}(k) - \lambda T \end{aligned} \tag{11}$$

By taking the average from time index 0 to time index $K-1$ on both sides of (11) and letting K goes to infinite, we have

$$\lim_{K\to\infty}\frac{Q_{i,j}(K)}{K} \geq \lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} d_{i,j}(k) - \lambda T \tag{12}$$

Thus, if the virtual queue $Q_{i,j}(K)$ is rate stable, we have

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} d_{i,j}(k) \leq \lambda T \tag{13}$$

This completes the proof. ■

By introducing the rate stable virtual queue $Q_{i,j}(k)$, the constraint (8) can be rewritten as:

$$\lim_{K\to\infty}\frac{Q_{i,j}(K)}{K}\leq 0, \forall i\in\mathbf{I}, \forall j\in\mathbf{J} \tag{14}$$

Taking advantage of the above relaxations and motivated by Lyapunov Optimization [28]–[30], we can further transfer the time-average problem to a one-step optimization problem so that the joint adaptation challenge can be resolved separately at each time slot. Before calculating the Lyapunov penalty, we first introduce a concatenated vector of the virtual queues as:

$$\Theta_j(k)\triangleq[Q_{1,j}(k), Q_{2,j}(k), ..., Q_{I,j}(k)] \tag{15}$$

Thus, the corresponding Lyapunov penalty function for every receiver can be calculated as:

$$P(\Theta_j(k))\triangleq\frac{1}{2}\sum_{i=1}^{I}Q_{i,j}(k)^2 \tag{16}$$

With (16), the Lyapunov penalty drift $\Delta(\Theta_j(k))$ is:

$$\Delta(\Theta_j(k))=E\{P(\Theta_j(k+1))-P(\Theta_j(k))|\Theta_j(k)\} \tag{17}$$

Enforcing the virtual queue constraints is equivalent to minimizing the drift penalty. Thus, we can turn the original maximization problem into a minimization problem so that maximizing the overall QoE with virtual queue constraint is equivalent to minimizing the following drift-plus-penalty term:

$$\sum_{j=1}^{J}\Big[\Delta(\Theta_j(k))+V_j(-\eta_j QoE_j(k))\Big] \tag{18}$$

where V_j is a non-negative weight to allow a trade-off between the virtual queue constraint and the objective of overall QoE maximization.

Therefore, original problem formulation (6) can be transformed to the one-step problem as follows:

$$\begin{aligned}\min_{\{r_k^{i,j},c_k^i\}}\quad & \sum_{j=1}^{J}\Big[\Delta(\Theta_j(k))-V_j(\eta_j QoE_j(k))\Big]\\ s.t.\quad & \tfrac{r_k^{i,j}}{c_k^i}\leq B_i^{up}(k), \forall i\in\mathbf{I}, \forall j\in\mathbf{J}\\ & \textstyle\sum_{i=1}^{I}\tfrac{r_k^{i,j}}{c_k^i}\leq B_j^{down}(k), \forall j\in\mathbf{J}\\ & r_k^{i,j}\in\mathbf{R}, c_k^i\in\mathbf{C}\end{aligned} \tag{19}$$

Furthermore, we can simplify the formulation of (19) by finding the upper bound of the penalty drift in (17).

$$\begin{aligned}&P(\Theta_j(k+1))-P(\Theta_j(k+1))\\ =&\frac{1}{2}\sum_{i=1}^{I}\Big[Q_{i,j}(k+1)^2-Q_{i,j}(k)^2\Big]\\ \leq&\frac{1}{2}\sum_{i=1}^{I}\Big[d_{i,j}(k)-\lambda T\Big]^2+\sum_{i=1}^{I}Q_{i,j}(k)\Big[d_{i,j}(k)-\lambda T\Big]\end{aligned} \tag{20}$$

Substituting (20) to (17), we can get:

$$\Delta(\Theta_j(k))\leq C_j+\sum_{i=1}^{I}Q_{i,j}(k)\Big[d_{i,j}(k)-\lambda T\Big] \tag{21}$$

where C_j is constant that always satisfies the below condition:

$$C_j\geq\frac{1}{2}\sum_{i=1}^{I}E\Big[(d_{i,j}(k)-\lambda T)^2|\Theta_j(k)\Big] \tag{22}$$

Thus, after the further Lyapunov optimization and the upper bound substituting, the original time-average problem is now simplified to the following one-step joint adaptation problem:

$$\begin{aligned}\min_{\{r_k^{i,j},c_k^i\}}\quad & \sum_{j=1}^{J}\Big\{\sum_{i=1}^{I}Q_{i,j}(k)\Big[d_{i,j}(k)-\lambda T\Big]-V_j(\eta_j QoE_j(k))\Big\}\\ s.t.\quad & \tfrac{r_k^{i,j}}{c_k^i}\leq B_i^{up}(k), \forall i\in\mathbf{I}, \forall j\in\mathbf{J}\\ & \textstyle\sum_{i=1}^{I}\tfrac{r_k^{i,j}}{c_k^i}\leq B_j^{down}(k), \forall j\in\mathbf{J}\\ & r_k^{i,j}\in\mathbf{R}, c_k^i\in\mathbf{C}\end{aligned} \tag{23}$$

C. SJA Algorithm for Joint Adaptation

The one-step problem in Section V-B still need to make decisions on the two separated action space, and it still need to deal with the coupling objective. To tackle these difficulties, we propose our SJA algorithm based on primal decomposition, which can obtain the near-optimal solution quickly on the server. We first rephrase the formula in (23) as follows:

$$\begin{cases}f_0(\mathbf{r}_j,\mathbf{c},k)=\sum_{i=1}^{I}Q_{i,j}(k)\Big[d_{i,j}(k)-\lambda T\Big]-V_j(\eta_j QoE_j(k))\\ f_1(\mathbf{r}_j,\mathbf{c},k)=\frac{r_k^{i,j}}{c_k^i}-B_i^{up}(k)\\ f_2(\mathbf{r}_j,\mathbf{c},k)=\sum_{i=1}^{I}\frac{r_k^{i,j}}{c_k^i}-B_j^{down}(k)\end{cases} \tag{24}$$

where $\mathbf{r}_j=[r_k^{1,j}, r_k^{2,j}, ..., r_k^{I,j}]^T$ and $\mathbf{c}=[c_k^1, c_k^2, ..., c_k^I]^T$.

Thus, (23) can be represented in the following format:

$$\begin{aligned}&\min_{\{\mathbf{r}_j\},\mathbf{c}_k}\sum_{j=1}^{J}f_0(\mathbf{r}_j,\mathbf{c},k),\\ s.t.\quad & f_n(\mathbf{r}_j,\mathbf{c},k)\leq 0, n=1,2\\ & r_k^{i,j}\in\mathbf{R}, c_k^i\in\mathbf{C}\end{aligned} \tag{25}$$

To cope with the coupling problem, as both bitrate and FEC code rate determine the delay metric, we transfer the original problem into one master problem and several subproblems based on the primal decomposition. We introduce a master agent to account for the information update and action aggregations as shown in (25). We then introduce J slave agents to be responsible for subproblem optimization within each

Algorithm 1 SJA Algorithm for joint loss and bitrate adaption

Input:
Total times: K; Clients: $\mathbf{I}, \mathbf{J}$;
Bandwidth capacity: $B_i^{up}(k)$,$\forall i \in \mathbf{I}$; $B_j^{down}(k)$, $\forall j \in \mathbf{J}$;
Bitrate requests: $\hat{r}_k^{i,j}$, $\forall i \in \mathbf{I}, \forall j \in \mathbf{J}$;
Channel's loss rate: $p_i(k)$, $\forall i \in \mathbf{I}, \mathbf{J}$;
Threshold: ζ; Maximum iterations: n;
Output:
Optimal bitrate and FEC code rate: $r_k^{i,j}, c_k^i$;
Initialize $Q_{i,j}(k)$, $i = 1, 2, ...I$, $j = 1, 2, ..., J$;
for $k = 1$ to K **do**
  Initialize $iteration = 0$, $\triangle = \infty$;
  while $iteration < n$ and $\triangle > \zeta$ **do**
    for all $i, \in \mathbf{I}, j \in \mathbf{J}$ **do**
      $r_k^{i,j} = \underset{\mathbf{r}}{inf}\{f_0 | f_n \leq 0, n = 1, 2\}$;
      Update $\mathbf{r}_j$ to the master agent;
    end for
    for all $i \in \mathbf{I}, j \in \mathbf{J}$ **do**
      $r_k^{i,j} = \underset{r \in \mathbf{R}}{\arg\min} |f_0^j(r_k^{i,j}) - f_0^j(r)|$;
      $c_k^i = \underset{c \in \mathbf{C}}{\arg\min} \sum_{j=1}^{J} f_0^j(c)$;
    end for
    $\triangle = \sum_{j=1}^{J} f_0^j(k) - f_0^j(k-1)$;
    $iteration++$;
  end while
  Output joint adaption solutions $\mathbf{r}$ and $\mathbf{c}$ for time slot k;
end for

iteration, and every slave agent simulates a receiver to solve the following subproblem with only linear constraints:

$$\mathbf{r}_j^* = inf\{f_0 | f_n \leq 0, n = 1, 2\} \tag{26}$$

We now design the algorithm for joint loss and bitrate adaptation as shown in Algorithm 1, which can be implemented on the server with slave agents running in parallel. The required inputs of state information at each time slot include network conditions such as bandwidth capacity and channels' loss rates, as well as receivers' bitrate requests. Also, the parameters consist of maximum iteration between master agent and slave agents at each time slot and the incremental extent of the objective, which are responsible for the stop criteria.

In each iteration, every slave agent will first perform the subproblem optimization step in (26) on behalf of the corresponding receiver. As only one variable exists in the subproblem, every slave agent will roll a one-step optimization. They first adopt the fixed FEC code rate from the server agent to obtain the optimal bitrate choices for every receiver. It is worth noting that the encoding type constraints are omitted by every slave agent so that the original subproblem can be relaxed and transformed into a linear constraint problem, thus reducing the complexity of the solution. After that, every slave agent will update their optimal solutions to the master agent responsible for aggregations. Without bitrate clustering, It will assign the best encoding bitrate type according to the optimal bitrate solutions from slave agents through minimum objective sacrifice. Similarly, we can generate an optimal FEC code rate for each sender by minimizing the overall penalty drift.

The iteration will stop until it reaches the maximum iteration setting or the extent of the overall performance improvement is insignificant. Opposite to DP-based approaches, we emphasize that SJA algorithm has the superiority in processing time with time complexity of $O(nIJ)$ since it significantly increases the scalability by translating the original problem to a master problem and several one-variable subproblems with linear constraints. Thus, the optimal decisions to encode on senders and to stream forwarding on the server can be effectively and efficiently executed to achieve the optimized overall QoE.

VI. Performance Evaluation

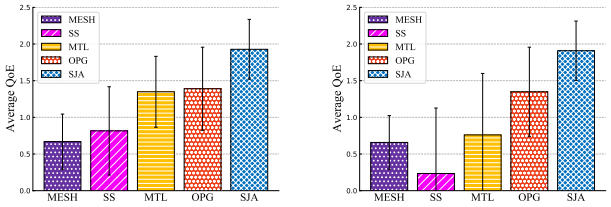
In this section, we compare the SJA framework with state-of-the-art MRVS approaches and evaluate their performance with extensive trace-driven experiments.

A. Methodology

Network Traces: To evaluate SJA and other state-of-the-art solutions under realistic network conditions, we create a corpus of network traces for more than 50 hours by combining two public datasets: (1) the FCC dataset [31] containing over 1 million throughput traces, each of which logs the average throughput over 2100 seconds. (2) the HSDPA dataset [32] on mobile throughput covers multiple usage scenarios such as buses, trains, and cities. To avoid trivial bitrate and FEC code rate selection, the synthesized corpus only considers the original traces with average throughput between 0.3 Mbps and 5 Mbps. Moreover, we collect the loss rate in two lossy network channels with an average of 1% and 2% loss rates.
Baselines: We consider the following methods as the baselines for comparison with SJA in MRVS scenarios:

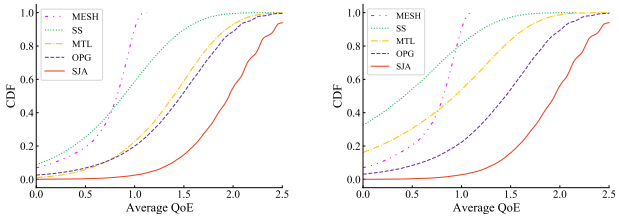
- *MTL:* An adaptive bitrate control algorithm of *MultiLive* for the multi-party realtime scenarios. It is a server-driven approach and comprehensively considers the uplink bandwidth and downlink bandwidth of all clients.
- *OPG:* An integrated *Oppugno* framework that achieves joint loss and bitrate adaption towards maximized QoE in realtime streaming services, which employs deep reinforcement learning algorithms on the receiver side.
- *SS:* Adopts *SVC* for sender-side video encoding and *SFU* architecture to select and forward streams with scalability.
- *MESH:* A simple version of *MTL* where *SVC* and *SFU* architectures are not applied.

Parameter Setting: Given the bandwidth traces, we set the available video bitrate as: $\{0.3, 0.5, 1.0, 2.0, 3.0, 5.0\}$ Mbps. We also set the available FEC code rates to range from 0.9 to 1.0. For the QoE weight setting, we choose $\{1, 1, 1, 2.5, 0.1\}$ and $\{1, 1, 1, 1.5, 0.25\}$ for perceptual quality, fluctuation, bitrate mismatch, loss damage, and streaming delay, respectively. Note that the loss damage weight 0.25 directly maps to the degradation of raw video quality after loss recovery, which is a non-negligible item. In that case, the two QoE preferences are delay-sensitive and loss-sensitive types. Besides, we set all



(a) The overall QoE over synthesized dataset with 1% loss (b) The overall QoE over synthesized dataset with 2% loss

Fig. 4. The overall QoE over synthesized dataset with real-world data



(a) CDF of QoE with 1% loss (b) CDF of QoE with 2% loss

Fig. 5. CDF of QoE metrics with real-world data

V_j as 1 to allow the balance between the time-average delay requirement and the objective of overall QoE maximization. Lastly, we set the maximum iteration in the SJA algorithm as 2 to ensure the fast response of the server-side orchestration.

Experiment Setup: To illustrate how effective our server-driven approach can achieve when facing multiple clients with different viewing preferences, we consider two cases with 5 clients. In the first case, every client in the meeting will request two videos from other clients simultaneously. More specifically, one video is in loss-sensitive type, and the other is in delay-sensitive type. Therefore, the client has different viewing preferences for different videos on its own screen. The detailed results are demonstrated in Section VI-B. In the second case, every client in the meeting will request four other videos; That is, every client can communicate with anyone in the video conferencing service. Moreover, we set three clients in the conference as delay-sensitive users, meaning these three clients will adopt delay-sensitive QoE models wherever the video is from. The remaining two clients in the meeting are loss-sensitive users, and they will adopt loss-sensitive QoE models. The experiment results are revealed in Section VI-C.

B. Performance with Real-world Data

In this section, we compare the performance of the SJA framework with other baselines. Based on our generated corpus, we evaluate the performance with two series of experiments with various loss conditions of 1% and 2% on average. Note that there are 5 clients and each viewer request two videos with loss-sensitive and delay-sensitive QoE preferences. The overall QoE achieved by SJA and other baselines are illustrated in Fig. 4, and the detailed CDF plots in Fig. 5. There are three key observations as follows.

Firstly, SJA is able to outperform all other baselines with higher average QoE, with 33.7% higher QoE than *MultiLive* in low loss rate conditions and 39.2% higher QoE than *Oppugno* in high loss rate conditions. This result confirms the effectiveness of SJA by integrating the loss and bitrate adaptation for overall QoE maximization. An interesting observation is that both *SS* and *MESH* are inefficient in dealing with the multi-party joint adaptation problem, achieving worse performance than other three frameworks as they lack server-driven bitrate orchestration and adaptive loss control mechanisms.

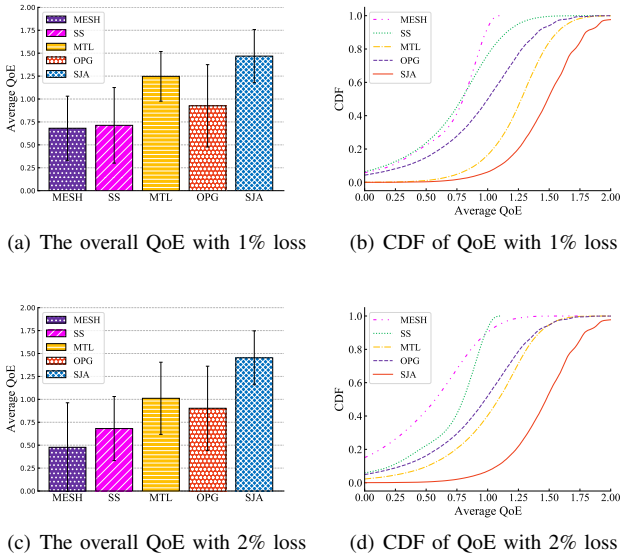
Secondly, SJA can achieve more stable and concentrated QoE scores than all other baselines. The CDF curves in the Fig. 5(a) and Fig. 5(b) show a smaller QoE variance of SJA than *MultiLive* and *Oppugno*. Moreover, the QoE scores of *Oppugno* are mainly concentrated within the range of 1.0 to 2.0, while *MultiLive* is even lower when the loss rate is relatively high. Instead, the QoE scores of SJA are mainly concentrated within the range of 1.5 to 2.5 in different loss conditions, reflecting that SJA can well mitigate the loss impact and can better handle the global streams orchestration with various loss and network throughputs, including the poor network condition with a large loss. The remaining two frameworks achieve quite concentrated but poor QoE performance, which infers that they fail to provide satisfactory QoE in multi-party realtime video streaming services.

The third observation is that the loss adaption is essential in multi-party realtime video streaming services. As shown in Fig. 4(a) and Fig. 4(b), there is a noticeable QoE drop for *MultiLive* as the loss rate increased, which indicates that *MultiLive* is quite sensitive to packet loss problems and insufficient to handle the large loss conditions even with server-side bitrate orchestration. Thus, this observation again confirms the superiority of our joint adaptation in MRVS services.

C. Performance of Generalization

In the experiments above, SJA performs well on the real-world data, while the realistic environment could be more complex with more viewing demands. To evaluate SJA's ability to generalize to more demanding conditions, we conduct two more experiments with various loss rates and more video requests. To be more specific, we evaluate the SJA framework and other baselines on the condition that every participant in the meeting room will request all four videos from other participants. Moreover, among the five participants, three serve as delay-sensitive users to attach great importance to streaming delay, while the other two participants are loss-sensitive and adopt loss-sensitive QoE models. We demonstrate the overall QoE achieved by SJA compared with other baseline methods and the detailed CDF plots in Fig. 6. After the generalization experiments, we have another two critical observations.

Firstly, the SJA framework performs better than all other baseline methods with higher average overall QoE scores and more concentrated QoE distributions in the more generalized multi-party realtime video streaming scenarios. As shown in Fig. 6(a) and Fig. 6(c), the average overall QoE of the SJA framework is at least 18.4% higher than other baseline methods in low loss rate conditions, and is at least 46.5%



(a) The overall QoE with 1% loss (b) CDF of QoE with 1% loss

(c) The overall QoE with 2% loss (d) CDF of QoE with 2% loss

Fig. 6. QoE performance of a more generalized case. There are 5 clients in the conferencing, each client request all 4 videos from others

higher than others in high loss rate conditions. Moreover, the QoE scores of SJA are mainly concentrated within the range of 1.25 to 1.75, reflecting that SJA can better mitigate the loss impact and solve the more challenging bandwidth allocation problem as the number of videos increases.

The second observation reveals the superiority of the server-driven joint adaptation design as the loss rate and the viewers' requested video number increase. Compared with the experiments in Section VI-B, the average overall QoE of all frameworks has been reduced. This result indicates that under the limited upload and download bandwidth, the video quality of users will decrease with the increase of requested video numbers. However, server-driven approaches have better resistance to video quality degradation as the requested video number rises. Opposite to Fig. 4(a), Fig. 6(a) shows that the decline of QoE of SJA and *MultiLive* are less than *Oppugno*, which obviously demonstrated the benefits of adopting server-driven strategies. Furthermore, in the high loss rate group shown in Fig. 6(c), the *MultiLive* solution with server-driven bitrate orchestration achieves similar QoE to the *Oppugno* solution with joint loss and bitrate adaption, while the server-driven SJA framework with joint loss and bitrate adaptation still performs the best. Thus, we can justify the importance of both the idea of server-driven and joint adaptation.

VII. Related Work

Previous ABR algorithm mainly studies how to dynamically adjust the bitrate to utilize the bandwidth of the current network and improve the QoE. According to the decision basis, existing ABR algorithms are mainly divided into four categories: *1) rate-based ABR:* rate-based approaches mainly focus on predicting future throughput and adjusting the bitrate based on predicted results. For example, Sun et al. [33] leveraged a data-driven approach to construct a Hidden-Markov-Model-based midstream predictor to model the stateful evolution of throughput. *2) buffer-based ABR:* buffer-based approaches adjust the bitrate by considering the buffer occupancy of clients. For example, Spiteri et al. [22] formulated bitrate adaptation as a utility-maximization problem and devised an online control algorithm called BOLA that uses Lyapunov optimization to minimize rebuffering and maximize video quality. *3) hybrid ABR:* hybrid methods usually consider multiple factors when deciding the future bitrate. For example, Yin et al. [26] propose a model predictive control algorithm that can optimally combine throughput and buffer occupancy information to outperform traditional approaches. *4) reinforcement-learning-based ABR:* rl-based methods utilize learning algorithms to achieve bitrate adaption. Mao et al. [1] pioneering trained a neural network model named Pensieve that selects bitrates for future video chunks based on observations collected by client video players. However, these studies on reliable TCP transmission do not consider the packet loss problem, which can cause picture distortion and seriously affect the user QoE.

Some previous studies have put forward efforts on multi-party realtime video streaming services. Hajiesmaili et al. [14] cast a joint problem of user-to-agent assignment and transcoding-agent selection, and devised an adaptive parallel algorithm to simultaneously minimize the cost of the service provider and the conferencing delay. Hu et al. [17] considered the multi-server architecture and formulated server selection as a geometric problem to propose fast heuristics with theoretical worst-case guarantees. Wu et al. [13] presented a cloud-assisted mobile video conferencing system to improve the quality and scale of multi-party mobile video conferencing, where the decentralized algorithm can decide the best paths of streams and the most suitable surrogates for video transcoding along the paths to utilize the limited bandwidth fully. Moreover, Wang et al. [12] considered global optimization that takes into consideration of different QoE factors, and designed an adaptive bitrate control algorithm for the multi-party scenario by modeling the many-to-many ABR selection problem as a non-linear programming problem. These MRVS solutions mainly focus on the optimization problem with cost-efficiency, service scalability, and low latency. However, they lack the consideration of the overall orchestration with handling the potential loss problems, which may lead to severe damage to perceptual video quality [34].

VIII. Conclusion

In this paper, we propose SJA, a server-driven joint loss and bitrate adaptation framework in multi-party realtime video streaming services towards maximized overall QoE. We first explored the importance of loss recovery and server orchestration in MRVS, and abstracted its service model to better tackle the challenges of selecting the best bitrate, FEC code rate, and streaming forwarding choices. Afterwards, we investigated the QoE model and mathematically formulated the problem, followed by proposing the SJA algorithm with close-to-optimal performance. Over a broad set of network conditions and QoE metrics, the SJA framework outperformed existing advanced solutions by $18.4\% \sim 46.5\%$.

REFERENCES

[1] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 197–210.

[2] F. Wang, C. Zhang, F. Wang, J. Liu, Y. Zhu, H. Pang, and L. Sun, "Deepcast: Towards personalized qoe for edge-assisted crowdcast with deep reinforcement learning," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1255–1268, 2020.

[3] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022," *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018.

[4] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *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*, 2017, pp. 183–196.

[5] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 495–511.

[6] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1208–1216.

[7] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "{SENSEI}: Aligning video streaming quality with dynamic user sensitivity," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 303–320.

[8] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong, and L. Song, "Perm: Neural adaptive video streaming with multi-path transmission," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1103–1112.

[9] X. Zuo, J. Yang, M. Wang, and Y. Cui, "Adaptive bitrate with user-level qoe preference for video streaming," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1279–1288.

[10] L. Sun, T. Zong, S. Wang, Y. Liu, and Y. Wang, "Towards optimal low-latency live video streaming," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2327–2338, 2021.

[11] D. Zhang, K. Shen, F. Wang, D. Wang, and J. Liu, "Towards joint loss and bitrate adaptation in realtime video streaming," in *2022 The IEEE International Conference on Multimedia and Expo (ICME)*, 2022, pp. 1–6.

[12] Z. Wang, Y. Cui, X. Hu, X. Wang, W. T. Ooi, Z. Cao, and Y. Li, "Multilive: Adaptive bitrate control for low-delay multi-party interactive live streaming," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 923–938, 2021.

[13] Y. Wu, C. Wu, B. Li, and F. C. Lau, "vskyconf: Cloud-assisted multi-party mobile video conferencing," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, 2013, pp. 33–38.

[14] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay design for multiparty cloud video conferencing," *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2760–2774, 2017.

[15] Zoom Corporation, "Meeting and phone statistics," https://support.zoom.us/hc/en-us/articles/202920719-Meeting-and-phone-statistics, 2021.

[16] J. Wang, G. Zhao, H. Xu, Y. Zhao, X. Yang, and H. Huang, "Trust: Real-time request updating with elastic resource provisioning in clouds," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 620–629.

[17] Y. Hu, D. Niu, and Z. Li, "A geometric approach to server selection for interactive video streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 840–851, 2016.

[18] Y. Yuan, W. Wang, Y. Wang, S. S. Adhatarao, B. Ren, K. Zheng, and X. Fu, "Vsim: Improving qoe fairness for video streaming in mobile environments," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1309–1318.

[19] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.

[20] Y. Chen, Q. Li, A. Zhang, L. Zou, Y. Jiang, Z. Xu, J. Li, and Z. Yuan, "Higher quality live streaming under lower uplink bandwidth: an approach of super-resolution based video coding," in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2021, pp. 74–81.

[21] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[22] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[23] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[24] T. Schierl, C. Hellge, S. Mirta, K. Gruneberg, and T. Wiegand, "Using h. 264/avc-based scalable video coding (svc) for real time streaming in wireless ip networks," in *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2007, pp. 3455–3458.

[25] J. Korhonen and P. Frossard, "Flexible forward error correction codes with application to partial media data recovery," *Signal Processing: Image Communication*, vol. 24, no. 3, pp. 229–242, 2009.

[26] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[27] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[28] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on average delays and queue size averages and variances in input-queued cell-based switches," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 1095–1103.

[29] N. Li, Y. Hu, Y. Chen, and B. Zeng, "Lyapunov optimized resource management for multiuser mobile video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1795–1805, 2018.

[30] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[31] Office of Engineering and Technology, "Raw data measuring broadband america 2021," https://www.fcc.gov/oet/mba/raw-data-releases, 2021.

[32] R. Haakon, V. Paul, G. Carsten, and H. Pål, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of IEEE MMSys*, 2013.

[33] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[34] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 1. IEEE, 1999, pp. 345–352.