Optimized Live 4K Video Multicast

Zhaoyuan He[1], **Changhan Ge**[1], **Wangyang Li**[1], **Lili Qiu**[1,2], **Peijie Li**[3], **Ghufran Baig**[1]
[1]Department of Computer Science, The University of Texas at Austin, Austin, TX, USA
[2]Microsoft Research Asia Shanghai, Shanghai, P.R.China
[3]Google, Austin, TX, USA

Abstract

4K videos are becoming increasingly popular. However, despite advances in wireless technology, streaming 4K videos over mmWave to multiple users is facing significant challenges arising from directional communication, unpredictable channel fluctuation and high bandwidth requirements. This paper develops a novel 4K layered video multicast system. We (i) develop a video quality model for layered video coding, (ii) optimize resource allocation, scheduling, and beamforming based on the channel conditions of different users, and (iii) put forward a streaming strategy that uses fountain code to avoid redundancy across multicast groups and a Leaky-Bucket-based congestion control. We realize an end-to-end system on commodity-off-the-shelf (COTS) WiGig devices. We demonstrate the effectiveness of our system with extensive testbed experiments and emulation.

1 Introduction

Motivation: With affordable ultra-high-definition (UHD) displays coming out and 4K contents becoming widely available, 4K video streaming has gained tremendous popularity in recent years. A 4K video frame has 3840x2160 pixels, which is $4\times$ resolution of standard full-high-definition (FHD) pictures. This yields several significant benefits: (i) 4K videos render finer details on the screen and allow users to see farther by improving the image depth [1]. (ii) more pixels blend colors more naturally and objects more realistically. These improvements contribute to immersive user experience, which is critical for virtual/augmented-reality (VR/AR) gaming.

The huge benefits of 4K videos have motivated researchers to develop innovative solutions improving 4K video rendering, compression and streaming. However, most of the current works focus on single user scenario. Delivering 4K content to multiple users remains less-studied though we observe an increasing need of 4K video multicasting. Living sports, concerts, conferences and gaming all require video streaming to multiple users simultaneously. For example, in VR/AR gaming or film-watching, multiple users gather at the same place and the game/content server renders and distributes the videos to user stations (STA). To reduce latency, the game/content servers may close to or even co-located with the base station or Wi-Fi access point (AP) [22, 8].

The above scenarios motivate us to explore efficient ways to transmit videos from a base station or AP to multiple users. Honoring the real-time requirement in gaming is especially challenging: supporting 30 frames per second (FPS) means finishing encoding, transmission and decoding within 33 ms (even without considering rendering cost). Given this high data requirement, millimeter-wave (mmWave) is appealing. However, it suffers large and unpredictable throughput fluctuation [42], making it hard to bear stable video quality. Moreover, serving multiple users becomes even more challenging as 60 GHz may not support all users at high data rates due to directional communication and rapid signal attennuation over the distance. To accommodate heterogeneous and dynamic wireless links across users, it is essential to adapt the resource allocation across users to optimize overall video quality.

Challenges: Realizing a live 4K video multicast system poses several significant challenges. Users having different signal-to-noise-ratios (SNR) are likely to receive different amounts of data. DASH [31], a standard unicast protocol for video streaming over Internet, divides a video into chunks encoded in multiple bitrates. Each client adapts its bitrate for video chunks according to the network condition. Directly applying DASH to multicast would significantly limit the sharing across users since only users of the same bitrate can share the content.

Second, to optimize video streaming, we seek a utility function that accurately translates the amount of received content to the video quality. The function should be tailored to a specific video coder, and should be general to support different videos.

Third, the video quality highly depends on the transmission strategy. How to schedule video transmissions to optimize video quality across multiple users with different channel conditions? This involves (i) how to put users into multicast groups, (ii) how to beamform to each multicast group, (iii) how to allocate time across multicast groups, (iv) how to map the time allocation into a practical packet level scheduler while avoiding redundancy across multicast groups, and (v) how to provide resilience and avoid congestion in multicast.

Fourth, mmWave channels fluctuate widely with the mobility at the receiver or environment. Such fluctuation is hard to predict and causes significant performance degradation in video streaming (*e.g.*, unable to send the frames at the selected rates, which causes decoding errors). We seek an approach that quickly adapts to the dynamic channel and gracefully compromise performance when the network condition degrades.

Our approach: Our end-to-end live 4K video multicast system addresses the above challenges in the following ways.

First, to effectively support video streaming to multiple users with diverse network conditions, we combine layered coding with rateless code. Layered coding efficiently supports multicast since the shared lower layer only needs to be transmitted once to all receivers, whereas conventional codec requires separate copies to be transmitted if the users request different resolutions. We use Jigsaw [2], a live 4K layered video coding implemented on COTS devices. While the original Jigsaw targets a single user, we address several new issues in multi-user scenario, such as cross-layer cross-user resource allocation and adaptation to unpredictable and diverse wireless losses at different users. To enhance resilience, we leverage Raptor code-based [28] source coding within each layer at the sender. Raptor code is a type of fountain code. The sender uses the Raptor code to generate a continuous stream of data for each video layer, and each receiver can decode the content as soon as it receives enough data. Raptor code allows efficient re-transmission to multiple receivers while significantly reducing the receiver feedback overhead.

Second, to optimize resource allocation, we need a utility function that can accurately map the amount of content received at each layer to a well-known video quality metric (*e.g.*, Structural Similarity (SSIM) [39]). We develop a Deep Neural Network (DNN) model for this purpose. We identify the input features that can capture the characteristics of different videos and allow us to develop a general model for diverse videos. Our methodology is general enough to support other video quality metrics, such as Peak signal-to-noise ratio (PSNR).

Next we design a novel framework that optimizes traffic allocation across different user groups. In particular, we perform multicast beamforming based on the channel state information (CSI) of multiple receivers for each possible multicast group. The resulting beams effectively improve the signal strength to various receivers. Then we optimize the transmission time allocation across different multicast groups and layers to maximize end-to-end video quality across all users.

The optimized time allocation assumes the traffic is a continuous stream that receiving more traffic means receiving more information. It holds when there is no data redundancy in traffic. But redundancy arises when a user belongs to multiple multicast groups. Interestingly, using Raptor-code can support efficient re-transmission while eliminating this redundancy. By encoding packet with Raptor code, we further map the time allocation to a practical packet scheduler, which assigns the packets to different multicast groups. Meanwhile, using Raptor code also builds resiliency against channel fluctuation. Our approach automatically re-transmits the data in the lower layer during channel degradation to support successful video frame decoding, albeit at a reduced resolution. In comparison, existing approaches cannot finish transmitting the data corresponding to the selected rate, which causes a loss of entire video frames. [20] reports a considerable degradation in existing video streaming approaches under mobile 5G channels.

We implement our system on commercial WiGig devices. To the best of our knowledge, this is the first layered video multicast streaming system over mmWave. Our testbed and emulation experiments show that our design yields above 0.975 SSIM and 43 dB PSNR when serving two users within the range of 3m, and above 0.94 SSIM and 35dB PSNR when serving up to eight users within the range of 16m. Moreover, under mobility, our system yields 0.008-0.068 SSIM and 1.5dB-6.4dB PSNR improvement when serving one single user, and 0.006-0.248 SSIM and 1.1dB-8.7dB PSNR improvement when serving three users. We also quantify the benefit of each component in our system.

To sum up, our contributions are as follow:

- We develop a DNN-based video quality model;
- We develop a cross-layer optimization algorithm to derive beamforming weights and time allocation across different multicast groups and layers. We further translate the optimized allocation into a practical packet-level scheduler;

- We implement an end-to-end system that addresses several practical challenges, including using rateless code for efficient retransmission and overlapping multicast groups, lightweight rate control, and adaptation to the dynamic channel. We demonstrate its effectiveness under various channel conditions (*e.g.*, mobile channels arising from client movement or environment changes).

2 Our Approach

2.1 Overview

We use layered coding for video multicast to accommodate dynamic channel conditions across different users. At a high level, the video codec partitions a video frame into multiple layers. Each multicast group is assigned ≥ 1 layers. The more layers received, the higher the video quality. Hence, we first develop a DNN model that determines the video quality based on the amount of traffic received at each layer. Then we propose an algorithm that optimizes time allocation across multicast groups to maximize the overall video quality across users. We further develop a holistic system that turns the optimization results into protocol configurations and addresses practical issues, including avoiding redundancy across different multicast groups, loss recovery, and rate control.

2.2 Layered Video Coding

Layered coding is attractive for video multicast since it allows users with different channel conditions to share layers. For example, when two users experience different channel conditions, they will request different videos in DASH [31]. In comparison, layered coding allows them to share the common low layers. Thus, the benefit of multicast increases with the number of users. Moreover, layered coding is robust to throughput fluctuation that the receiver decodes a lower quality video when throughput drops and a higher quality video when throughput improves. This feature is desirable for mmWave, whose throughput tends to fluctuate rapidly.

We use Jigsaw [2] as the underlying layered video coding since it supports live 4K video on COTS devices. It first divides a video frame into non-overlapping 8x8 blocks of pixels. The base layer 0 consists of averages of pixel values in 8x8 blocks, which yields 512x270 resolution. Next, it divides the 8x8 blocks into 4x4 blocks and assigns each 4x4 block with a pixel value difference between the average of 4x4 block and the average of 8x8 block, forming layer 1. Similarly, it forms layer 2 representing the difference of pixel values in 2x2 blocks, and layer 3 representing the difference in 1x1 blocks.

Following the terminology in [2], layers 1-3 contain multiple sub-layers. For example, layer 1 has the difference values $D(i,j,k)$, where (i,j,k) is the k-th 4x4 block within the (i,j)-th 8x8 block. The k-th sublayer in layer 1 consists of $D(i,j,k)\ \forall\ i,j$. Similarly, we define sublayers in other layers.

2.3 Video Quality Model

We then design a model to determine the video quality based on the amount of data received at each layer. The video quality model is necessary since it is used as an optimization objective in the resource allocation optimization problem (Sec. 2.4). The models we use to fit the data are Linear Regression, Support Vector Machine (SVM), and a lightweight DNN.

We generate dataset using 6 uncompressed 4K videos (4096×2160 resolution) of 1000 frames and YUV420 format from Derf's collection under Xiph [40]. We include videos with various motion and spatial locality. 3 videos are high richness (HR) and 3 are low richness (LR), where HR and LR differs by variance in the Y values. We randomly split the dataset by 7 : 3 for training and testing with no overlapping.

We feed the DNN with the following input : (i) the number of packets received at each layer, (ii) SSIM value if everything up to the i-th layer has been received completely, and (iii) SSIM value of the blank frame. We include (ii) since different video frames have diverse SSIM values in each layer. For example, LR videos have higher SSIM in the base layer. We include (iii) because different video frames have distinct differences from a blank frame. Both (ii) and (iii) can be computed efficiently. The output is the SSIM of the corresponding video frame. For data collection, we feed different fractions of each video layer to a video decoder and use *FFmpeg* to compute the SSIM.

From Table 1, we find DNN performs the best since it is flexible to capture the non-linear relationship. Our DNN consists of 5 fully connected layers, as shown in Fig.1(a). Each layer has the same number of input and output features (*i.e.*, $in\_features = 9$ and $out\_features = 9$). A Sigmoid activation layer follows each fully connected layer. Finally, the model ends with a linear layer with 9 $in\_features$ and 1 $out\_features$ to generate an estimated SSIM value. We use Adam [16] as the optimizer and MSE as the loss function. The model is trained using 500 epochs with a batch size of 128.

Fig. 1(b) compares the estimated and actual SSIM, where the center, lower and upper cap of the error bar denote the average, lowest and highest accuracy, respectively. Our model produces high estimation accuracy across all layers. The DNN inference time is around 500μs on our WiGig devices.

Method	SVM	Linear Regression	DNN
MSE	0.0524	0.0231	2.4313×10^{-5}

Table 1: Video quality model based on different methods.

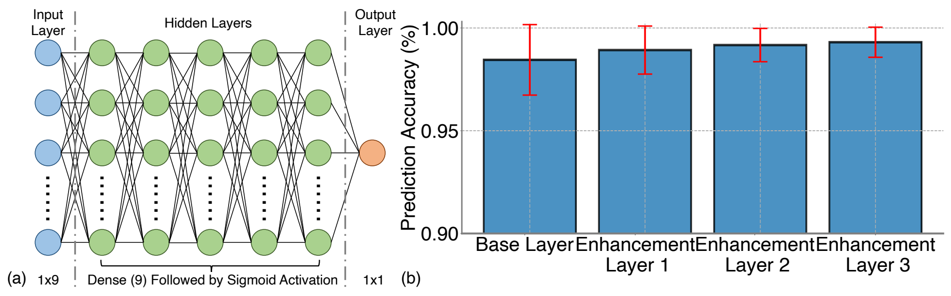

Figure 1: Video quality model: (a) DNN Structure (b) Accuracy

MCS	Sensitivity (dBm)	Iperf3-UDP (Mbps)	MCS	Sensitivity (dBm)	Iperf3-UDP (Mbps)	MCS	Sensitivity (dBm)	Iperf3-UDP (Mbps)
0	-78	×	5	-62	×	9.1	-57	×
1	-68	300	6	-63	1050	10	-55	1850
2	-66	550	7	-62	1250	11	-54	2100
3	-65	720	8	-61	1580	12	-53	2400
4	-64	850	9	-59	×	$\geq$ 12.1	$\geq$-51	×

Table 2: Modulation Coding Scheme (MCS), receiver sensitivity and UDP throughput. × not supported for data traffic on QCA6320

2.4 Optimizing Transmission Strategy

Built on top of our video quality model, we seek to determine a transmission strategy that optimizes the end-to-end video quality across all users. For N clients, we enumerate all possible user groups. For each user group, we use beamforming to maximize the minimum Received Signal Strength (RSS) to the group. We can map the RSS to the data rate using a standard lookup table as shown in Table 2. We omit the groups whose throughput is below a threshold to speed up computation. Given the set of multicast groups and the maximum data rate delivered to each group, our goal is to derive the time allocation across different multicast groups and video layers. We formulate the problem as follow:

$$\begin{aligned} \max_{T_{G,j}} \quad & \sum_i Q(D_{i,1}, D_{i,2}, D_{i,3}, D_{i,4}) - \lambda \sum_i D_{i,j} \\ \text{subject to} \quad & D_{i,j} = \sum_{i \in G} T_{G,j} R_G, \forall i, j \\ & \sum_{G,j} T_{G,j} \leq \frac{1}{FR} \end{aligned} \tag{1}$$

where $Q(.)$ denotes the video quality given the amount of data received for each layer, $D_{i,j}$ denotes the data volume that user i receives at layer j, $T_{G,j}$ denotes the time allocated to multicast group G for sending layer j, R_G denotes the multicast data rate to user group G, and FR is the frame rate.

The objective reflects our goal of maximizing the video quality, which is a function of the amount of data received at each layer. We use DNN to learn $Q(.)$ to map the amount of data received at each layer to video quality (*e.g.*, SSIM) as described in Sec. 2.3. We also include a penalty term defined by the total traffic weighted by λ, which aims to minimize the amount of traffic when the video quality is the same. λ is small to ensure that the video quality is the primary objective, and total traffic is only used to break ties.

The first constraint reflects that the total amount of data delivered to a user i at each layer j is the sum of data transmitted in all groups involving the user i for that layer. The second constraint enforces that the total transmission time across all groups and layers is no more than the time budgeted for the video frame. For example, to reach 30 FPS, any video frame should be transmitted within $1/30$ seconds. Note that we do not enforce the size restriction for each layer because a user may receive some redundant data if it participates in multiple multicast groups. However, optimizing our objective will automatically minimize redundancy.

2.5 Multicast Beamforming

To support the above optimization, we need to determine the throughput for each multicast group. We use beamforming to improve the RSS for a given multicast group. We then map the RSS to the data rate, which is the input of Problem 1.

Given an AP with N_t antennas and a STA with N_r antennas, the frequency-domain received signal can be expressed as:

$$\mathbf{y} = [\mathbf{F}]^{1\times N_\mathrm{t}}[\mathbf{h}]^{N_\mathrm{t}\times N_\mathrm{r}}[\mathbf{W}]^{N_\mathrm{r}\times 1}\mathbf{s} + \mathbf{n} \tag{2}$$

where $[\cdot]^{m\times n}$ is the dimension of matrices, $\mathbf{h}$ is the channel matrix, $\mathbf{W}$ and $\mathbf{F}$ is the combiner of STA and precoder of AP, respectively, $\mathbf{s}$ is the normalized signal.

In IEEE 802.11ad [23], AP performs sector-level-sweeping (SLS) to determine the proper precoder $\mathbf{F}$. Suppose the STA only has one quasi-omnidirectional antenna. The AP operates with a set of K predefined beams formed by $[\mathbf{F}]^{N_\mathrm{t}\times K}$, whose radiation patterns jointly cover the azimuth plane. The AP first broadcasts beacon frames precoded with different columns in $\mathbf{F}$. Then, STA measures the sequence of *per-beam RSS*, denoted as $[\mathbf{r}]^K$, where $r^k = |\mathbf{F}^k\mathbf{h} + \mathbf{n}|^2$ and k is the beam index. The STA feeds back AP the best beam index, *i.e.*, $\arg\max\limits_{k\in K}[\mathbf{r}^k]^K$. Current WiGig cards (*e.g.* Sparrow+ [12]) work with at most $K = 128$ due to limited on-board memory, and non-trivial beam training overhead [32]. Moreover, the AP cannot exhaust the precoders since the search space increases exponentially with N_t and phase shifter bits M, namely M^{N_t}. Hence, the selected codebook may not be optimal.

CSI-based beamforming can achieve better performance than SLS. Following Adaptive Codebook Optimization (ACO) [24]), we can estimate the CSI $\mathbf{h}$ of each receiver. Given the CSI, we can optimize the unicast codebook as $\mathbf{h}^H/||\mathbf{h}||$.

To multicast to N users, we find $\mathbf{F}$ that satisfies:

$$\max_{\mathbf{F}} \min[\mathbf{r}]^{1\times N} = |\mathbf{F}[(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N)]| \tag{3}$$

where $\mathbf{h}_i$ is the channel matrix of i-th receiver.

The Max-Min problem in Eq. 3 is NP-hard [19]. To speed up the optimization, we observe that finding the maximum sum RSS of a group of receivers can be optimized efficiently using singular value decomposition (SVD) [19]. Specifically, suppose we want to maximize the $\sum \mathbf{r} = \|\mathbf{H}\mathbf{F}'\|^2$, where $\mathbf{H} \triangleq [\mathbf{h}_1; \mathbf{h}_2; ...\mathbf{h}_n]$. Decomposing $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$, the beamforming weight F is the first column of $\mathbf{V}$. While the Max-Sum is different from our original goal, it is a good heuristic.

Based on the optimized $\mathbf{F}$, we derive the RSS and select the MCS accordingly. The corresponding data rate of each multicast group is fed to Problem 1 for optimization.

2.6 Packet Scheduling

The optimization output from Problem 1 indicates how much traffic to transmit to each multicast group at each layer. We use it to guide packet scheduling. The new issue we should address is how to avoid data redundancy. For example, when the output says sending 30%, 40%, and 50% of layer 2 to multicast groups 1, 2, and 3, which portion of layer 2 should we send to ensure that the users involved in multiple multicast groups receive minimal redundant packets? A simple approach is to send packets directly from the layered encoder like Jigsaw, which requires a careful packet assignment to different groups to minimize redundancy. Moreover, in the presence of packet loss, which is common in WiGig links, we also need feedback for loss recovery.

Source coding can simplify the design and significantly reduce data redundancy and feedback overhead. We use Raptor code [28], a fast rateless code, to further encode Jigsaw's encoded data into a continuous stream of data. Unlike the original symbols, any two Raptor coded symbols carry different information and the receiver just needs to accumulate enough Raptor coded symbols for successful decoding. We choose Raptor Code because any reception of K packets

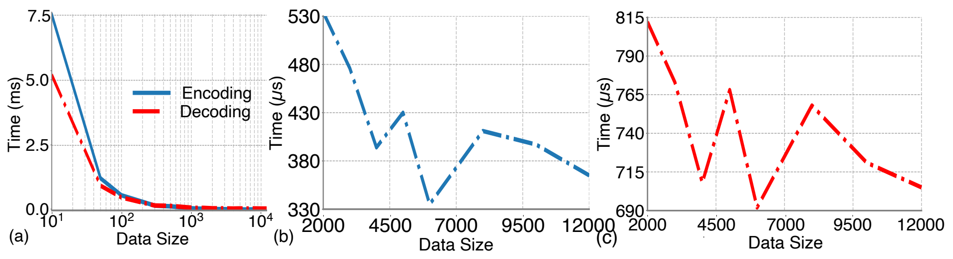

Figure 2: RaptorQ encoding and decoding time changes with the packet size. (a) Overall trend. (b) Detailed trend of encoding. (c) Detailed trend of decoding

can be decoded with a high probability. Specifically, receiving $K + h$ symbols yields a decoding probability of $1 - 1/256^{h+1}$, where K is the number of symbols to code and h is the number of extra symbols received. By applying Raptor code, we can significantly simplify transmission strategy – the sender continuously generate data stream until the receivers can decode. We port the Rust-based RaptorQ [3] to C++ and integrate it with Jigsaw.

To use rateless code, we need to determine the symbol size and the number of symbols for coding. We use a sublayer in Jigsaw as a coding unit. The symbol size affects the encoding and decoding time. As shown in Fig. 2, the encoding and decoding time both initially decrease with the symbol size and then increase. We set the symbol size to 6000B, which is close to the shortest encoding and decoding time.

Each sublayer contains 20 symbols. Data within the same sublayer is equivalent, each adding new information related to the same sublayer. However, data across different sublayers is different. Therefore, instead of tracking at packet-level, we should track the sublayer-level reception status. This is much lighter to maintain. Two issues remain to be addressed: (i) how to map traffic allocation across layers and multicast groups in Problem 1 to the allocation across coding groups since only packets within the same coding group are equivalent and packets belonging to different coding groups carry different information, and (ii) how to maintain the group information.

To address (i), we need to split the transmissions across multiple coding groups within the layer to maximize the decoded information. Specifically, denote $S(G, j)$ as the transmission size allocated to the multicast group G and layer j, $sss(G, i, j)$ as the transmission size allocated to the coding group i in layer j at group G, and $ss(u, i, j)$ as the total decoded traffic by user u for the coding group i in layer j. We formulate the following optimization problem:

$$\begin{aligned} &\max \sum_i \sum_u ss(u,i,j) \\ &\text{subject to } \forall G, j : \sum_i sss(G,i,j) \leq S(G,j) \\ &\forall u : ss(u,i,j) = \begin{cases} size(i,j), & \sum_{u \in G} sss(G,i,j) \geq size(i,j) \\ 0, & o.w. \end{cases} \end{aligned} \tag{4}$$

Our goal is to find $sss(G, i, j)$ that maximizes the total received traffic across all coding groups and users. The first constraint indicates the total traffic received across all coding groups within a layer j and multicast group G is bounded by the total allocation for the layer. The second constraint indicates the total traffic to be decoded is either the complete coding group when we receive at least the coding group size amount of data or 0 otherwise. This is due to the nature of source coding, which requires receiving enough data to decode the coding group. The second constraint can be converted into an integer linear constraint using an indicator variable. Let $u = ss(u, i, j)$, $S = size(i, j)$, $v = \sum_{u \in G} sss(G, i, j)$. We can convert the second constraint into $0 \leq S - v + ku \leq kS$ and $u \in \{S, 0\}$. When $v \geq S$, $S - v$ is negative, u has to be S; when $v < S$, $S - v$ is positive, u has to be 0 to satisfy the inequity. We solve it using a greedy heuristic where we assign traffic to the coding groups in an increasing order; within the same coding group, we assign it to the multicast groups in an increasing order of group id until all receivers across each group get the complete data.

To address (ii), we let the receiver report the number of packets it receives for each sublayer and each multicast group. Upon receiving the feedback, the sender takes the difference between the number of packets it has sent and the number of the packets reported by the receiver. Let P denote the difference between the two numbers. The sender will transmit additional P packets as a makeup for the packet loss. Using Raptor code, we reduce the feedback from packet-level to sublayer-level, which is easier to track. Moreover, to support live video streaming and coding, the feedbacks and all retransmissions should finish within 33 ms for 30 FPS videos.

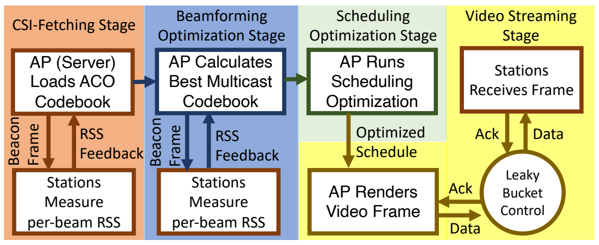
CSI-Fetching Stage
AP (Server) Loads ACO Codebook
Beacon Frame
RSS Feedback
Stations Measure per-beam RSS
Beamforming Optimization Stage
AP Calculates Best Multicast Codebook
Beacon Frame
RSS Feedback
Stations Measure per-beam RSS
Scheduling Optimization Stage
AP Runs Scheduling Optimization
Optimized Schedule
AP Renders Video Frame
Video Streaming Stage
Stations Receives Frame
Ack
Data
Leaky Bucket Control
Ack
Data

Figure 3: System Workflow

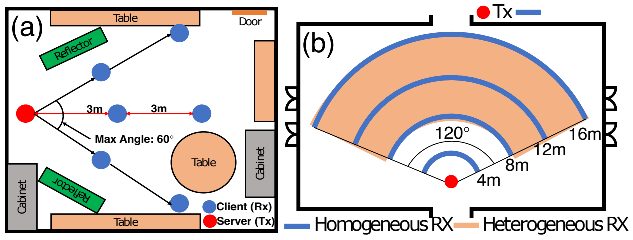

Figure 4: (a) user placement in testbed experiment. (b) user placement in emulation evaluation

2.7 Leaky Bucket-Based Rate Control

Transmission Control Protocol (TCP) is widely used for unicast congestion control and loss recovery. However, our system targets multicast and uses Raptor code to generate newly coded packets for loss recovery instead of retransmitting the lost packets. Therefore, we use User Datagram Protocol (UDP). Our rate control problem is unique because it needs to support multicast and the priorities of different layers (*i.e.*, first ensure lower layers are received).

To meet the requirements, we use Leaky-Bucket [4]-based rate control. For each multicast group, we have a credit that specifies the maximum bytes the sender can send for a given time. The sender periodically increments the credit based on the desired sending rate of the group. It may send new data if it has enough credit. Upon each packet transmission, the sender decrements the credit by the packet size.

There are two critical parameters in the Leaky Bucket: the average credit filling rate and maximum credit the bucket can hold at a time. We set the former to the expected throughput. To limit the delay, we set the latter to a small value (*e.g.*, 10 packets) that still sustain high throughput.

In the beginning, the expected throughput is determined by the UDP throughput of the selected MCS. To keep up with the time-varying channel and avoid overhead, we let the receiver periodically measure the link bandwidth based on the difference in the arrival time of 100 data packets and feedback it to the sender. Moreover, for improved channel conditions, the packets used for bandwidth estimation should not be subject to the rate control and should be sent back to back. Hence, these packets are more likely to miss due to congestion. Since losses in the lower layers are lethal to the video quality, we use the packets from the highest layer for bandwidth measurement. The sender uses the bandwidth reported by the receiver during the previous video frame transmission to control the sending rate of a new video frame.

2.8 Adapting to Dynamic Channel

As mentioned in Sec. 2.5, CSI is required to optimize beamforming. We use the framework developed in ACO [24] and X-array [38] to estimate CSI based on SLS RSS feedback.

We have performed testbed experiments in static scenarios. For mobile cases, the patched firmware cannot stably dump SLS RSS while high data traffics exist. Therefore, we record the RSS traces measured at each receiver to compute CSI. We then use the CSI trace to drive emulation, which runs the same code as the testbed except that the data traffic is sent over emulated links. The trace-driven emulation allows us to compare various algorithms under the same channel condition, which is hard to enforce in mobile scenarios.

3 Testbed Implementation

We build our video-streaming system on 4 Acer Travelmate P658 laptops with Intel i7-6500U CPU, NVIDIA GeForce 940M graphics card, and Qualcomm QCA6320-based 802.11ad network card. One laptop serves as an AP (server, video encoder), and three serve as STAs (clients, video decoder). The GPU on our platform performs similar to Qualcomm Adreno 650, a mainstream mobile GPU currently used on Oculus Quest 2 VR headset and Samsung S20 series.

3.1 System Workflow

As shown in Fig. 3, our system begins with fetching CSI using ACO [24]. Then, it calculates the multicast beamweights and feeds the estimated throughput of each multicast group to the scheduling optimizer. The optimization stage only takes a few milliseconds with multi-core computation. It can be performed every 100ms (*i.e.*, ACO beacon interval) and incorporated with 802.11ad/ay [23, 9] perfectly. Loading the optimized schedule and beamweights, the AP then calls network interface command *WMI_SET_SELECTED_RF_SECTOR_INDEX* (cmd_id: 0x9A3) to enforce multicast beams and calls *HW_SYSAPI_FORCE_MCS* (cmd_id: 0x900 ut_subtype_id: 0x6) to set MCS accordingly. Then it sends out video data at the rate according to the UDP throughput corresponding to the specified MCS (see Table 2). Note that the combined measured overhead of triggering the two commands is $\approx 25\mu$s, which is only 0.075% of the per-frame transmission time for 30 FPS and 0.15% for 60 FPS. To adapt to dynamic channel (*e.g.*, channel degradation), the STAs continuously monitors the packet arrival rate and feeds back to the AP. The AP uses the rate fed back to adjust its sending rate using the leaky bucket-based rate control.

3.2 Pseudo Multicast

A natural way to implement multicast is to let the WiGig card send multicast traffic. However, QCA6320 does not offer API to change MCS for multicast traffic. Multicast traffic can only use MCS 1 (sub-300 Mbps). To bypass this limitation, we associate one STA with the AP as a regular receiver and set all the other STAs to the monitor mode, which allows them to capture the data traffic not destined to them. This approach effectively achieves multicast while supporting any MCS. Another benefit is that the regular receiver can still enjoy MAC layer retransmissions and binary backoff in CSMA.

3.3 Throughput

As shown in Table 2, QCA6320 chipset only implements MCS from 0 to 12 excluding 5, 9, and 9.1 in 802.11ad [7]. Existing works [33, 38, 37] use RSS sensitivity table in [7] to map the RSS to MCS and data rate. We use the same table to determine the MCS of a multicast group based on the RSS. But the throughput we feed to the resource optimization is the measured UDP throughput as listed in the 3rd column of Table 2, which considers the PHY/MAC overhead. Each value is the average of 20 times 10s *Iperf3* trials.

4 Performance Evaluation

In this section, we thoroughly evaluate the designed system.

4.1 Evaluation Methodology

We implement our multicast video streaming system in both testbed and emulator. Both run the same video encoder, decoder, scheduler, source coding, and rate control. The only difference between the testbed and emulation experiments is that data is transmitted over the WiGig links, and beamforming is performed on phased arrays in the testbed, while there is no data sending over the air in the emulator. These two methodologies are complementary since the testbed evaluates the performance under realistic channel conditions while emulation allows us to consider using more extensive topologies. We use 2 HR videos and 2 LR videos from the dataset described in Sec. 2.3. Each of them is 30FPS and lasts at least 5 minutes.

4.2 Testbed Evaluation

We perform testbed evaluation indoor as shown in Fig.4(a). Unless otherwise specified, we use the following default configuration: the sender uses optimized multicast beamforming; all 3 clients are 3m away from the sender with their angular spacing randomly selected between 0 and 30°, *i.e.*, the maximum angular spacing (MAS) (*i.e.*, the spacing from the leftmost to the rightmost station) is 30°. We use high-richness videos and perform 10 random runs.

4.2.1 Impact of Beamforming

We compare the following beamforming schemes: (i) optimized multicast beamforming (Sec. 2), (ii) pre-defined multicast beam, (iii) optimized unicast beamforming, and (iv) pre-defined unicast beam. They all use the same optimization framework in Section 2.4 to optimize the schedule, and the only difference is the data rate fed to the optimization, which is derived based on the RSS. As expected, (i) > (ii) > (iii) > (iv). (i) generates multi-lobe beam pattern that covers multiple users at the same time, and performs the best. (ii) under-performs (i) due to the use of predefined beams, but out-performs (iii) and (iv) due to the use of multicast.

The STAs are 3m apart from the AP and the MAS between the users is 60° as shown in Fig.4(a). The results are shown in Fig. 5, where the lines on the box from the top to the bottom are the max, 1st quartile, median, 3rd quartile and min. In all cases, optimized multicast beamforming performs the best. Its benefit increases with the number of users. For example, the SSIM improvements over pre-defined multicast, optimized unicast, and pre-defined unicast are 0.012, 0.016, and 0.038 in 2 users; and 0.021, 0.023, and 0.045 in 3 users, respectively. The PSNR improvements are 2.5

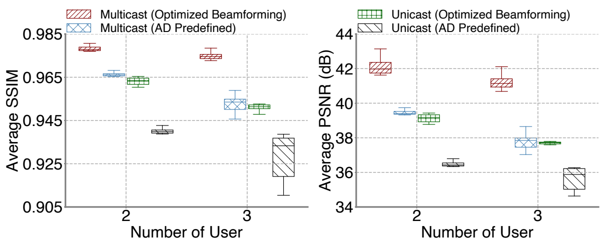

Figure 5: Testbed result for different number of user. Distance: 3m; Maximum angular spacing (MAS): 60°. (a) SSIM. (b) PSNR.

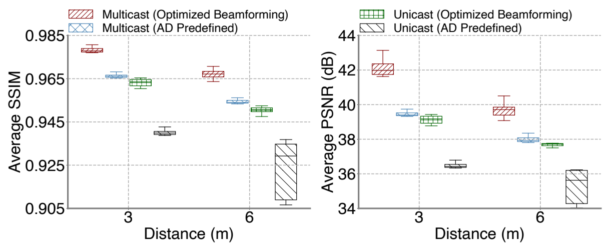

Figure 6: Testbed result as we vary the distance between server and clients. # Users: 2; MAS: 30°. (a) SSIM. (b) PSNR.

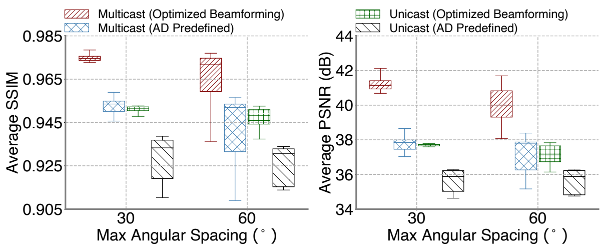

Figure 7: Testbed comparison of different maximum angular spacing (MAS) between the clients. # Users: 2; Distance: 3m. (a) SSIM. (b) PSNR.

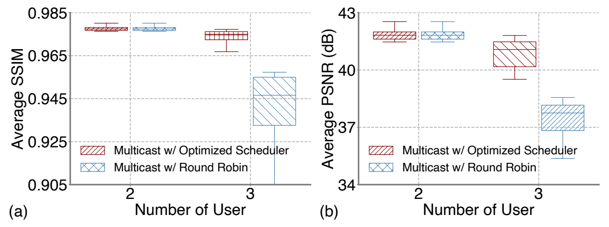

Figure 8: Testbed comparison between optimized scheduler and round robin. Distance: 3m; MAS: 60°. (a) SSIM. (b) PSNR.

dB, 2.9 dB, and 5.6 dB in 2 users; and 3.2 dB, 3.3 dB, and 5.4 dB in 3 users, respectively. 3 dB PSNR improvement means that the video quality doubles. Hence, the optimized multicast beamforming has a large advantage over the other methods. The variance of pre-defined unicast among 3 users is large since the best pre-defined beams may or may not steer towards the receiver. The user in the beam direction receives high video quality while others' video quality degrades significantly.

Impact of distances: Fig. 6 compares the performance as we vary the distance between the AP and STAs in testbed. There are 2 users, and their MAS is 30°. The SSIM at 3m are 0.976, 0.965, 0.963, and 0.939 for optimized multicast, pre-defined multicast, optimized unicast,and pre-defined unicast, respectively, while the SSIM at 6m are 0.966, 0.955, 0.951 and 0.924, respectively. Due to the layered video coding and schedule optimization, the video quality degrades gracefully with the increasing distance. Among different beamforming schemes, the optimized multicast beamforming continues to be the best: it outperforms the other schemes by 0.011-0.042 in terms of SSIM and by 1.8-5.6 dB in terms of PSNR.

Impact of maximum angular spacing (MAS): We place 2 users at 3m and vary the MAS. Fig. 7 shows that the optimized multicast beamforming yields 0.018-0.048 improvement in SSIM and 3-6 dB improvement in PSNR. It continues to out-perform the other alternatives across all different MAS. Moreover, MAS has little impact on unicast performance but affects multicast performance as we expect.

4.2.2 Impact of Schedule

Fig. 8 compares our scheduling with the round-robin [25] scheduling, which enumerates all possible user groups and uses round-robin to schedule across different user groups (*i.e.*, the sender transmits to each group for 1 ms and then uses the round-robin to select the next group to transmit for 1 ms, and so on). As we can see, our scheduling performs the same as the round-robin for 2 users because there is only one multicast group for 2 users. However, our scheduling outperforms the round-robin by 0.03 in SSIM and 3.2 dB in PSNR for 3 users because our approach allocates the transmission time across different user groups by explicitly considering their link quality and the impact of transmissions on video quality.

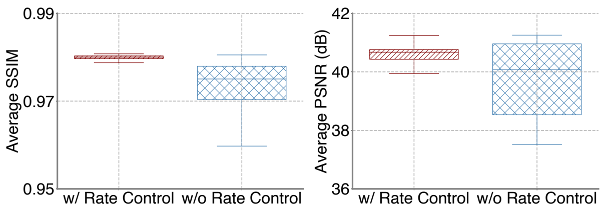

Figure 9: Testbed comparison between with and without rate control. # Users: 3; Distance: 3m; MAS: 60°; Beamforming: optimized multicast beamforming. (a) SSIM. (b) PSNR.

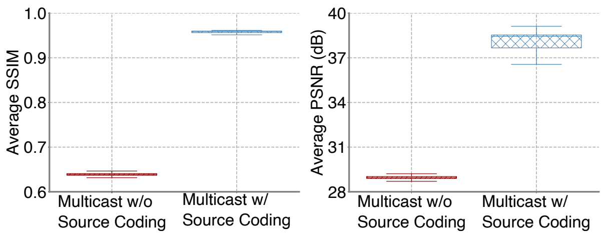

Figure 10: Testbed comparison between with and without source coding. # Users: 3; Distance: 3m; MAS: 60°; Beamforming: optimized multicast beamforming. (a) SSIM. (b) PSNR.

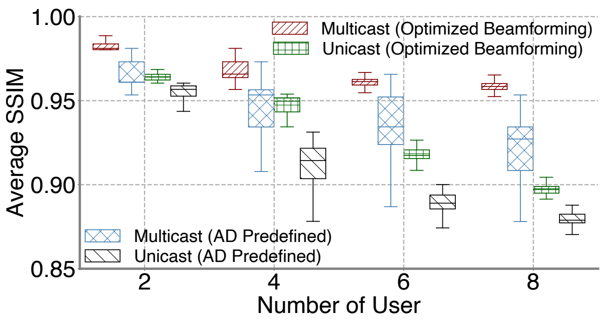

Figure 11: Emulation comparisons of different # users and beamforming. Distance: 8-16m; MAS: 120°.

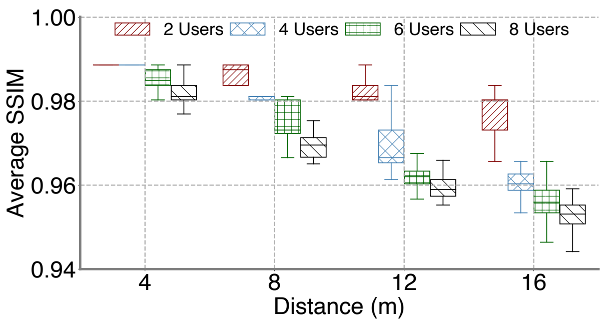

Figure 12: Emulation comparison of different distances between server and clients. MAS: 120°.

4.2.3 Impact of Rate Control

Fig. 9 shows the performance of our rate control approach. Without rate control, the AP sends packets to the driver continuously until the kernel's queue is full. This triggers packet drop and lead to low quality for several frames. This therefore lowers the SSIM for 0.01 and PSNR for 1.3 dB comparing with the case with rate control. Moreover, it yields larger variance in video quality since its performance fluctuates with random queue drops. With our leaky bucket-based rate control, we can reduce packet drops at the kernel and maintain high SSIM across all frames over different runs.

4.2.4 Impact of Source Coding

Fig. 10 compares the performance between with and without source coding. The former outperforms the latter by 0.32 in SSIM and 9.5 dB in PSNR. The result shows that multicast without source coding has much worse performance and higher variance due to inefficient retransmission to multiple receivers and redundancy arising from multicast groups with shared receivers.

4.3 Emulation Evaluation

Then, we evaluate our design using emulation. We first consider static cases. We use a lidar scanner to reconstruct 3D model of a meeting room and feed it to Wireless Insite [13], a popular commercial 3D ray-tracer widely used by the research community [14, 15], to generate realistic wireless channel. We place the users in two different ways: clients are randomly placed at a fixed distance (4m, 8m, 12m or 16m) from the server, or clients are randomly distributed between 8m and 16m from the server as shown in Fig.4(b). We perform 100 random runs for each configuration and show the aggregated results. Next, we evaluate mobile scenarios with CSI traces collected from testbed while moving the clients or environment. Note that there is no data sending over the air in the emulator, so no extra packets will be dropped over the channel. In this case, the video quality might be slightly higher than that in testbed under the same experimental setting. For brevity, we only show SSIM in this section.

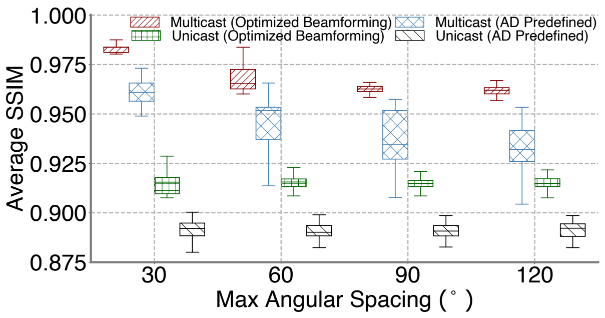

Figure 13: Emulation comparison of different MAS between clients. # Users: 6; Distance: 12m.

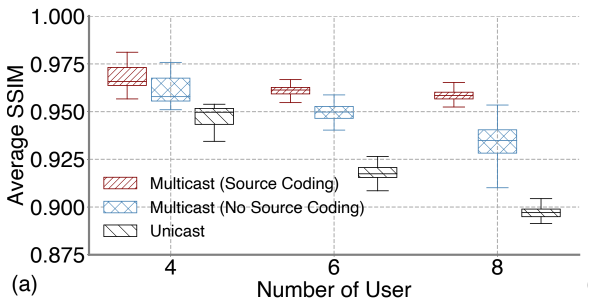

Figure 14: Emulation comparisons of with and without source coding. Distance: 8-16m; MAS: 120°.

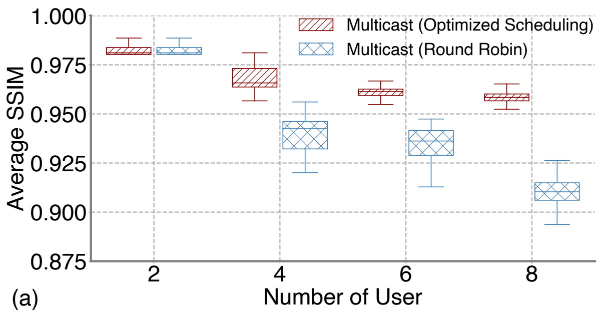

Figure 15: Emulation comparisons of different scheduling algorithms. Distance: 8-16m; MAS: 120°.

4.3.1 Impact of Beamforming

Varying the number of clients: Fig. 11 shows the result when a varying number of clients are randomly placed between 8m and 16m from the server and the MAS between users is 120°. The multicast with optimized beamforming improves over pre-defined multicast, optimized unicast, and pre-defined unicast by 0.010, 0.013, and 0.025 in 2 user case; 0.025, 0.025, and 0.055 in 4 users case, 0.03, 0.04, and 0.075 in 6 users scenario, 0.035, 0.06, and 0.083 in 8 users case, respectively.

The emulation result further verifies our conclusion from the testbed experiment that the multicast benefit increases with the number of users since a single transmission can satisfy more users in multicast. The exact amount of improvement depends on the difference in SNR across the users as the bottleneck user limits the multicast throughput. Moreover, multicast also improves fairness since multicast tries to transmit packets that benefit multiple users.

Impact of distance: We further evaluate the impact of the distance between the server and users. As shown in Fig. 4(b), we place the users at 4m, 8m, 12m, and 16m with 120° MAS. The server operates with optimized beamforming. As shown in Fig. 12, the performance of optimized multicast beamforming slightly fluctuates with the variation of the distance and the number of users. The average SSIM difference across number of users is 0.01, 0.015, 0.025, and 0.03 at 4m, 8m, 12m, and 16m, respectively. As we have concluded in the testbed evaluation, the average SSIM difference between the number of users increases with the distances because of our layered video coding and schedule optimization.

Impact of maximum angular spacing (MAS): Fig. 13 compares the performance when 6 users are placed at 12m from the sender and their MAS is varying. In all cases, optimized or pre-defined multicast beamforming outperforms optimized or pre-defined unicast beamforming. Multicast performs the best when the MAS is small since it can concentrate the beam towards closely spaced users. In comparison, two unicast schemes perform similarly across different MAS as unicast is always directed to the target receiver and not affected by the locations of other receivers.

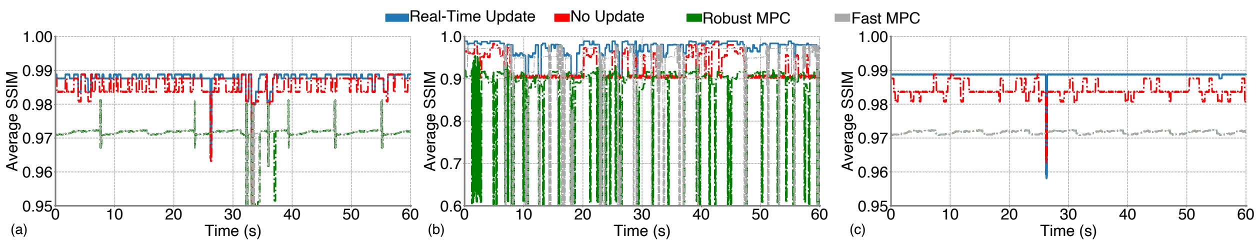

Figure 16: Emulation result of serving 1 receiver when (a) receiver is moving under High RSS ($\geq -61dBm$), (b) receivers are moving under Low RSS ($< -61dBm$), and (c) environment is moving.

4.3.2 Impact of Source Coding

Next, we evaluate the impact of source coding by enabling or disabling source coding. In both cases, we use optimized multicast beamforming and scheduling. Fig. 14 shows the video quality for 4, 6, and 8 users that are randomly placed between 8m and 16m from the server. As we can see, our source coding helps to remove redundancy and improves the SSIM by around 0.005-0.025. The emulation result draws the same conclusion as our testbed evaluation.

4.3.3 Impact of Scheduling

Fig. 15 compares our scheduling with round-robin scheduling while both uses optimized multicast beamforming. Same as Fig. 8, when there are 2 users, there is only one multicast group, and there is no difference in the two scheduling algorithms. Our scheduler improves over the round-robin by 0.029, 0.030, and 0.052 in SSIM under 4, 6, and 8 users scenarios, respectively. As expected, the importance of scheduling increases with the number of users.

4.3.4 Trace-driven Mobile Experiment

As explained in Sec.2.8, due to hardware limitations, we evaluate mobile scenario using the trace-driven method, which allows a fair comparison between different approaches under the same condition. The first type of mobile trace is obtained from moving receivers. While the server is broadcasting ACO beacon frames, two people hold the laptops (serving as clients) and walk randomly for a minute, and the clients measure CSI. The CSI measurements from different clients are synchronized with the server's timestamp. Since the beacon interval is 100ms [23], we have 10 CSI measurements per second. The second type of mobile trace is obtained from moving environment. While the server is broadcasting ACO beacon frames, two people walk randomly between the server and receivers to disturb wireless signals. The sender adapts beamforming, time allocation and packet schedule based on the each data point in CSI traces. We call it Real-time Update.

We first compare the approach that uses the beamforming, time allocation, and schedule computed at the beginning of the experiment, but does not adapt to the dynamic channel. We call this approach No Update. Then, we compare our approach with Adaptive bitrate (ABR) algorithms. [20] implements 7 ABR algorithms. [36] reports Robust MPC [41] and Fast MPC are two of the best ABR algorithms for live video streaming. Therefore we use these as our baselines. The two baselines make bitrate decisions by solving an optimization problem of the QoE for the next n chunks (*e.g.*, $n = 5$). Although they perform well under static network conditions, they cannot sustain a good video quality in mobile environment. The reason is that DASH streaming typically uses standard video codecs, such as H264, HEVC, and VP9. The above codecs fail to decode subsequent frames if the current frame is not decoded due to packet loss. In comparison, our layered coding is more resilient to channel degradation. Moreover, the existing RPC based methods are tailored for unicast streaming.

We first compare all approaches under a single user. We use the RSS sensitivity of MCS 8 (-61dBm) as the separation of the high RSS and low RSS. Fig.16 (a) shows the video quality of a single moving user under high RSS. On average, the Real-time Update out-performs the No Update, the Robust MPC, the Fast MPC by 0.008, 0.018, 0.016 in SSIM. Under high RSS, the two MPCs have similar performance for the single user unicast, but they are worse than the Real-time Update.

The video quality of a single moving user under low RSS is shown in Fig.16 (b). The Real-time Update out-performs the No Update, the Robust MPC, the Fast MPC by 0.008, 0.021, and 0.068 in SSIM, respectively. As the network condition worsens, the Robust MPC and Fast MPC have higher degradation than the Real-time Update. The benefit of the Real-time Update over the No Update mainly comes from our adaptation to the dynamic channel. The benefit over other two baselines comes from the effectiveness of our layered coding.

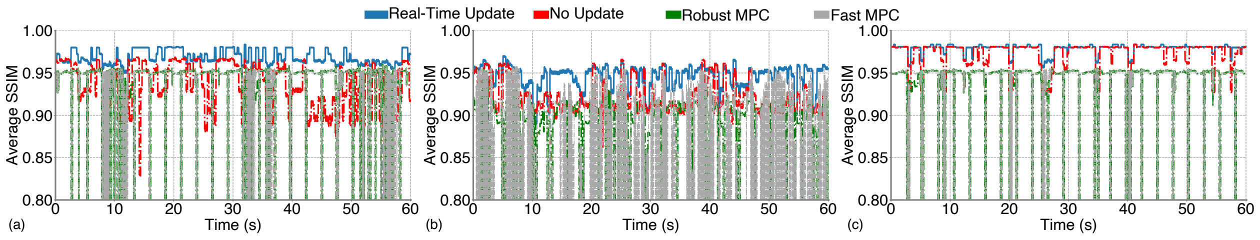

Figure 17: Emulation result of serving 3 receivers when (a) two receivers are moving under High RSS ($\geq -61dBm$), (b) two receivers are moving under Low RSS ($< -61dBm$), and (c) environment is moving.

Fig.16(c) shows that the Real-time Update out-performs the No Update, the Robust MPC, the Fast MPC by 0.004, 0.017, 0.017 in SSIM under the moving environment where two people are walking randomly between the server and receiver. Since the network condition for the single user unicast remains stable, there is no difference between the two baselines and no significant video quality degradation. However, the Real-time Update is still the best one among all approaches.

Then we compare these approaches when serving three users. We randomly move two of the receivers and keep the other receiver static. The Robust MPC and Fast MPC need to allocate time resources to each user for unicast, leading to a lower video quality than our multicast approach. Fig.17(a) shows the video quality of serving three receivers under high RSS. The Real-time Update can maintain high video quality when the SNR is still high: its SSIM is above 0.95 all time. In contrast, the video quality under the No Update fluctuates widely over time since the initial beamforming and schedule cannot work well at new clients' locations. The two MPC schemes have worse performance because they have coarse-grained bitrate options and cannot adapt within a group of pictures (GoP). When there is a large throughput drop, they still attempt to send at a higher rate, which cannot finish before the deadline. On average, the Real-time Update out-performs the No Update, the Robust MPC, the Fast MPC by 0.034, 0.059, and 0.064 in SSIM. The higher gain is due to the combined benefits from our layered coding and multicast.

Fig.17(b) shows the video quality of serving three receivers under low RSS. The Real-time Update out-performs the No Update, the Robust MPC, the Fast MPC by 0.026, 0.087, and 0.248 in SSIM, respectively. Since the SSIM degrades non-linearly with the decreasing throughput, the benefit of the Real-time Update over the No Update reduces. As the network condition worsens, the Robust MPC and Fast MPC have larger degradation than Real-time Update and No Update due to the limitations in ABR. Note that the lowest SSIM in Fig.17(b) is higher than that in Fig.16(b) because the former has at least one static receiver that does not suffer degradation, whereas the latter has only one receiver that experiences degradation.

Fig.17(c) shows the video quality of serving three receivers under the moving environment. The Real-time Update outperforms the No Update, Robust MPC, and Fast MPC by 0.006, 0.055, and 0.056 in SSIM. As the moving people change the environment, the network condition fluctuates. We see that the No Update is not affected much since the receivers are static, and it uses the same beamforming and time allocation as calculated at the beginning. Although the benefit over No Update gets smaller, our approach for three users yields a larger improvement over the two MPC schemes than the case of serving one user due to the additional multicast benefit.

5 Related Work

Video Streaming over WiGig: Given the large bandwidth in mmWave, streaming uncompressed video is a key application for WiGig. Choi *et al.* [6] propose a link adaptation policy that minimizes the total allocated resources by assigning different amount of resources to different data bits of a pixel. He *et al.* [11] encode an uncompressed video into multiple descriptions using RS coding. The video quality improves as more descriptions are received. [6, 11] use unequal error protection to protect different bits of a pixel-based importance. Shao *et al.* [27] compress difference pixel values using run-length encoding. It is challenging to parallelize the run-length codes since different pixels are unknown in advance. Singh *et al.* [29] partition adjacent pixels into different packets and adapt the number of pixels based on estimated throughput, but it suffers poor video quality when throughput drops.

Layered video multicast: LVMR [18] deploys an error recovery scheme using smart retransmission and adaptive playback point to address the network congestion and heterogeneity problem for layered video multicast. RLC [34] is a receiver-driven congestion control algorithm that is TCP-friendly and suitable for continuous data transfer. Still, it lacks an optimization of resource allocation for each multicast group. SAMM[35] uses congestion feedback to adjust the number of generated layers and the bit rate of each layer. It reduces the network congestion but ignores the redundancy of the received packets by different users when multicast is applied. Layered video multicast has been considered in the past (*e.g.*, [10, 17, 26, 2]), but it is still challenging for 4K video streaming under 60GHz WLAN. 4K layered coding has rarely been used commercially because of the high computational cost [2]. And the performance of mmWave fluctuates widely with the mobility of the transmitter, receiver, or environment.

Beamforming: Many algorithms have been developed to maximize the beamforming gain. Among them, [21] is one of the most related work: it develops multicast beamforming algorithms for 60GHz WLAN. It begins with the finest beams to ensure reachability and replaces the finer beams with wider ones to cover more clients if the utility improves. Our work uses CSI to optimize beamforming, which will yield higher throughput than the iterative search in [21].

Resource allocation and scheduling: There has been lots of work on resource allocation and packet scheduling in both single-hop and multi-hop wireless networks. [5, 30] survey some of existing resource allocation and scheduling approaches. Our work goes beyond theoretical analyses and considers several practical protocol design issues, such as mapping flow-level allocation to packet scheduling, avoiding redundancy and congestion, and system implementation.

Differences from the related work: Our work is the first end-to-end 4K video multicast system over COTS mmWave devices. It leverages layered coding to accommodate heterogeneous clients, optimizes resource allocation and packet schedules to achieve efficiency, and uses Raptor code and Leaky-Bucket-based rate control to avoid redundancy and congestion. Through system implementation, we also uncover limitations of commodity mmWave hardware and develop several effective approaches to address them (*e.g.*, video streaming based on beamforming and RSS feedback and pseudo multicast).

6 Conclusion

This paper develops an end-to-end live 4K video multicast system. It encompasses several significant components: modeling video quality, optimizing traffic allocation, packet scheduling, using source coding to address redundancy issues, leveraging rate control to avoid congestion, and adaptation to dynamic channels. We address specific research challenges and construct an effective system that can be used for video-related research. Our emulation and testbed experiments demonstrate its effectiveness.

References

[1] J. Archer. 4k tvs: 9 reasons you should buy one – and 9 more why you shouldn't, 2015.

[2] G. Baig, J. He, M. A. Qureshi, L. Qiu, G. Chen, P. Chen, and Y. Hu. Jigsaw: Robust live 4k video streaming. In *MobiCom*, pages 1–16, 2019.

[3] C. Berner. Rust-based implementation of raptorq, 2021.

[4] M. Butto, E. Cavallero, and A. Tonietti. Effectiveness of the'leaky bucket'policing mechanism in atm networks. *IEEE Journal on selected areas in communications*, 9(3):335–342, 1991.

[5] M. Chitnis, P. Pagano, G. Lipari, and Y. Liang. A survey on bandwidth resource allocation and scheduling in wireless sensor networks. In *NBiS*, 2009.

[6] M. Choi, G. Lee, S. Jin, J. Koo, B. Kim, and S. Choi. Link adaptation for high-quality uncompressed video streaming in 60-ghz wireless networks. *IEEE Transactions on Multimedia*, 2016.

[7] I. C. S. L. S. Committee et al. Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11^*, 2007.

[8] Edge. How edge compute is enabling better and more affordable gaming experiences, 2019.

[9] Y. Ghasempour, C. R. C. M. da Silva, C. Cordeiro, and E. W. Knightly. Ieee 802.11ay: Next-generation 60 ghz communication for 100 gb/s wi-fi. *IEEE Communications Magazine*, 55(12):186–192, 2017.

[10] C. Guo, Y. Cui, D. W. K. Ng, and Z. Liu. Multi-quality multicast beamforming with scalable video coding. *IEEE Transactions on Communications*, 66(11):5662–5677, 2018.

[11] Z. He and S. Mao. Multiple description coding for uncompressed video streaming over 60ghz networks. In *Proceedings of the 1st ACM workshop on Cognitive radio architectures for broadband*, pages 61–68. ACM, 2013.

[12] Q. Inc. 3gpp lte propagation channel models, 2017.

[13] R. Inc. Wireless insite 3d wireless prediction software, 2020.

[14] Z. Jiang, S. Zhou, Z. Niu, and C. Yu. A unified sampling and scheduling approach for status update in multiaccess wireless networks. In *INFOCOM*, pages 208–216. IEEE, 2019.

[15] K. Joshi, D. Bharadia, M. Kotaru, and S. Katti. Wideo: Fine-grained device-free motion tracing using rf backscatter. In *NSDI*, 2015.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] R. Kuschnig, I. Kofler, and H. Hellwagner. An evaluation of tcp-based rate-control algorithms for adaptive internet streaming of h. 264/svc. In *MMSys*, pages 157–168. ACM, 2010.

[18] X. Li, S. Paul, and M. Ammar. Layered video multicast with retransmissions (lvmr): Evaluation of hierarchical rate control. In *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, volume 3, pages 1062–1072. IEEE, 1998.

[19] M. J. Lopez. Multiplexing, scheduling, and multicasting strategies for antenna arrays in wireless networks. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF ELECTRICAL ENGINEERING AND . . . , 2002.

[20] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, et al. A variegated look at 5g in the wild: performance, power, and qoe implications. In *SIGCOMM*, pages 610–625, 2021.

[21] S. Naribole and E. Knightly. Scalable multicast in highly-directional 60-ghz wlans. *TON*, 25(5):2844–2857, 2017.

[22] L. Networks. Limelight, 2022.

[23] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer. Ieee 802.11ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper]. *IEEE Communications Magazine*, 52(12):132–141, December 2014.

[24] J. Palacios, D. Steinmetzer, A. Loch, M. Hollick, and J. Widmer. Adaptive codebook optimization for beam training on off-the-shelf ieee 802.11 ad devices. In *MobiCom*, pages 241–255, 2018.

[25] R. V. Rasmussen and M. A. Trick. Round robin scheduling–a survey. *European Journal of Operational Research*, 188(3):617–636, 2008.

[26] 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*, 17(9):1103–1120, Sept 2007.

[27] H.-R. Shao, J. Hsu, C. Ngo, and C. Kweon. Progressive transmission of uncompressed video over mmw wireless. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–5. IEEE, 2010.

[28] A. Shokrollahi. Raptor codes. *IEEE transactions on information theory*, 52(6):2551–2567, 2006.

[29] H. Singh, J. Oh, C. Kweon, X. Qin, H.-R. Shao, and C. Ngo. A 60 ghz wireless network for enabling uncompressed video communication. *IEEE Communications Magazine*, 46(12), 2008.

[30] I. Sousa, M. P. Queluz, and A. Rodrigues. A survey on qoe-oriented wireless resources scheduling. *CoRR*, abs/1705.07839, 2017.

[31] T. Stockhammer. Dynamic adaptive streaming over http: standards and design principles. In *MMSys*, 2011.

[32] S. Sur, I. Pefkianakis, X. Zhang, and K.-H. Kim. Practical mu-mimo user selection on 802.11 ac commodity networks. In *MobiCom*, 2016.

[33] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra. Beamspy: Enabling robust 60 ghz links under blockage. In *NSDI*, pages 193–206, 2016.

[34] L. Vicisano, J. Crowcroft, and L. Rizzo. Tcp-like congestion control for layered multicast data transfer. In *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies.* IEEE, 1998.

[35] B. J. Vickers, C. Albuquerque, and T. Suda. Source-adaptive multilayered multicast algorithms for real-time video distribution. *IEEE/ACM transactions on Networking*, 8(6):720–733, 2000.

[36] C. Wang, J. Guan, T. Feng, N. Zhang, and T. Cao. Bitlat: bitrate-adaptivity and latency-awareness algorithm for live video streaming. In *Proceedings of the 2019 ACM on Multimedia*, pages 2642–2646, 2019.

[37] S. Wang, J. Huang, and X. Zhang. Demystifying Millimeter-Wave V2X: Towards Robust and Efficient Directional Connectivity Under High Mobility. In *MobiCom*, 2020.

[38] S. Wang, J. Huang, X. Zhang, H. Kim, and S. Dey. X-array: approximating omnidirectional millimeter-wave coverage using an array of phased arrays. In *MobiCom*, 2020.

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[40] XIPH. Video dataset.

[41] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *SIGCOMM*, volume 45(4), pages 325–338. ACM, 2015.

[42] R. Zhao, T. Woodford, T. Wei, K. Qian, and X. Zhang. M-cube: a millimeter-wave massive mimo software radio. In *MobiCom*, pages 1–14, 2020.