Optimized Live 4K Video Multicast Streaming on Commodity WiGig Devices

Zhaoyuan He[†], Changhan Ge[†], Wangyang Li[†], Lili Qiu[†,‡], Peijie Li[§], Ghufran Baig[†]
[†]*The University of Texas at Austin*, Austin, TX, United States of America
[‡]*Microsoft Research Asia*, Shanghai, People's Republic of China
[§]*Google*, Austin, TX, United States of America
{zyhe,chge,wangyangli,lili,ghufran}@cs.utexas.edu, peijieli@google.com

2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-8605-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDCS60910.2024.00108

***Abstract*—The popularity of 4K videos is on the rise. However, streaming such high-quality videos over mmWave to several users presents significant challenges due to directional communication, fluctuating channels, and high bandwidth demands. To address these challenges, this paper introduces an innovative 4K layered video multicast streaming system. We (i) develop a video quality model tailored for layered video coding, (ii) optimize resource allocation, scheduling, and beamforming based on the channel conditions of different users, and (iii) design a streaming strategy that integrates fountain code to eliminate redundancy in multicast groups, coupled with a Leaky-Bucket approach for congestion control. We implement our system on Commodity-Off-The-Shelf (COTS) WiGig devices and demonstrate its effectiveness through comprehensive testbed and emulation experiments.**

***Index Terms*—Video streaming, Millimeter-wave, Multicast, Layered coding**

I. Introduction

With affordable Ultra-High-Definition (UHD) displays coming out and 4K content 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 Reality (VR) and Augmented Reality (AR) gaming.

The huge benefits of 4K videos have motivated researchers to develop innovative solutions to improve 4K video rendering, compression, and streaming. However, most of the current works focus on single-user scenarios. Delivering 4K content to multiple users remains less-studied though we observe an increasing need for 4K video multicasting. Live 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 (STAs). To reduce latency, the game/content servers may be close to or even co-located with the base station or WiFi Access Point (AP) [2, 3].

The above scenarios motivate us to explore efficient ways to transmit videos from a base station or AP to multiple users. It is well-known that real-time 4K gaming and streaming are demanding tasks. Supporting 30 Frames Per Second (FPS) means performing encoding, transmission, and decoding within 33 ms, even without considering rendering cost. Given such a challenge, millimeter-wave (mmWave) communication is appealing since its ultra-wide bandwidth enables a 10 Gbps-level capability [4] and there are Commodity-Off-The-Shelf (COTS) WiGig 60 GHz devices. However, realizing a live 4K video multicast system faces several significant challenges as follows.

Challenges: First, users having different Signal-to-Noise Ratios (SNRs) are likely to receive different amounts of data. This disparity becomes even more severe in mmWave scenarios due to the rapid signal attenuation over the distance and the adoption of directional communication. DASH [5], a standard unicast protocol for video streaming over the 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, mmWave suffers from large throughput fluctuations due to mobility at the receiver or environment [4, 6], which are hard to predict and cause significant performance degradation in video streaming (*e.g.*, unable to transmit the frames at the selected data rates, hence decoding errors). We seek an approach that swiftly adapts to the dynamic channel conditions and gracefully compromises the performance when the network condition degrades.

Third, the video quality highly depends on the transmission strategy, *i.e.*, how to schedule video resources and transmission schedules to optimize video quality across multiple users with different channel conditions. The first thing that matters is to build 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 enough to support videos of different richness. Apart from this, we also need to address (i) how to assign users into multicast groups, (ii) how to beamform towards 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.

2575-8411/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDCS60910.2024.00108

Our approach: We address the above challenges as follows.

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 [7], 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 scenarios, 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 [8] 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 support the resource allocation optimization, we develop a lightweight Deep Neural Network (DNN)-based utility function that accurately maps the amount of content received at each layer to Structural Similarity Index (SSIM) [9], a widely used video quality metric. We identify the input features that can capture the characteristics of different videos, which allows us to develop a general model for diverse videos. Our method 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 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 such that receiving more traffic means receiving more information. It holds when there is no data redundancy in traffic. However, 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 packets 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. Whenever the channel degrades, our approach automatically re-transmits the data in the lower layer 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 the loss of entire video frames [10].

We implement our system on commercial WiGig devices. To the best of our knowledge, this is the first layered video multicast streaming system over commodity mmWave devices. Our extensive 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 improvement when serving one single user, and 0.006-0.248 SSIM improvement when serving three users.

To summarize, our contributions are as follows:

- We develop a lightweight 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 to avoid redundancy, lightweight rate control, and adaptation to the dynamic channel. We demonstrate its effectiveness through extensive testbed and emulation experiments.

Ethics Statement: No personally identifiable information (PII) is used. This work does not raise any ethical concern.

II. Related Work

Video Streaming over WiGig: Given the large bandwidth in mmWave, streaming uncompressed video is a key application for WiGig. Choi *et al.* [11] 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.* [12] encode an uncompressed video into multiple descriptions using RS coding. The video quality improves as more descriptions are received. [11, 12] use unequal error protection to protect different bits of a pixel-based importance. Shao *et al.* [13] 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.* [14] 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 [15] 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 [16] is a receiver-driven congestion control algorithm that is TCP-friendly and suitable for continuous data transfer. Still, it lacks an optimization for resource allocation across multicast groups. SAMM [17] 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.*, [18, 19, 20]), 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 [7]. 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 numerous works on resource allocation and packet scheduling in both single-hop and multi-hop wireless networks. [22, 23] 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 WiGig 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).

III. Our Approach

A. 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.

B. 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 [5]. 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 since 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.

TABLE I
Video quality model based on different methods.

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

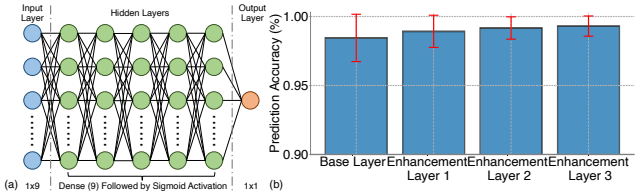


Fig. 1. Video quality model: (a) DNN Structure (b) Accuracy

We use Jigsaw [7] as the underlying layered video coding since it supports live 4K video via unicast 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 [7], 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.

C. 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. III-D). 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 [24]. 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 without overlap.

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 the differences between different video frames and a blank frame vary a lot. 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.

TABLE II
MCS, RECEIVER SENSITIVITY AND UDP THROUGHPUT. × MEANS "NOT SUPPORTED FOR DATA TRAFFIC ON QCA6320 CHIPSET"

MCS	Sensitivity (dBm)	iPerf3-UDP (Mbps)	MCS	Sensitivity (dBm)	iPerf3-UDP (Mbps)	MCS	Sensitivity (dBm)	iPerf3-UDP (Mbps)	MCS	Sensitivity (dBm)	iPerf3-UDP (Mbps)
0	-78	×	4	-64	850	8	-61	1580	11	-54	2100
1	-68	300	5	-62	×	9	-59	×	12	-53	2400
2	-66	550	6	-63	1050	9.1	-57	×	$\geq$ 12.1	$\geq$-51	×
3	-65	720	7	-62	1250	10	-55	1850	N/A		

From Table I, 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 [25] 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 $500\mu s$ on our WiGig devices.

D. 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 II. 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. III-C. 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.

E. Multicast Beamforming

To support the 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 CSI matrix, $\mathbf{W}$ and $\mathbf{F}$ are the STA's combiner and AP's precoder, respectively, $\mathbf{s}$ is the normalized signal, and $\mathbf{n}$ is the noise.

In IEEE 802.11ad [26] and 802.11ay [27], 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.*, $\arg\max_{k \in K} [\mathbf{r}^k]^K$. Current WiGig cards (*e.g.* Sparrow+ [28]) support at most $K = 128$ due to limited on-board memory, and non-trivial beam training overhead [29]. 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. Existing approaches, such as ACO [30] and 2ACE [4], can estimate the CSI matrix of each receiver. Given the CSI, we can optimize the unicast codebook as $\mathbf{h}^\mathrm{H}/||\mathbf{h}||$.

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 [31]. 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) [31]. Specifically, suppose we want to maximize $\sum \mathbf{r} = \|\mathbf{HF}'\|^2$, where $\mathbf{H} \triangleq [\mathbf{h}_1; \mathbf{h}_2; ...\mathbf{h}_n]$. Decomposing $\mathbf{H} = \mathbf{U\Sigma V}'$, the beamforming weight $\mathbf{F}$ 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 Modulation and Coding Scheme (MCS) accordingly. The corresponding data rate of each multicast group is fed to Problem 1 for optimization.

F. 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 be sent 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 [8], a fast rateless code, to further encode Jigsaw-encoded data into a continuous data stream. Unlike the original symbols, any two Raptor coded symbols carry different information and the receiver just needs to accumulate enough coded symbols for successful decoding. We choose Raptor Code because any reception of K packets 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. Raptor code significantly simplifies our transmission strategy – the sender continuously generate data stream until the receivers can decode. We port the Rust-based RaptorQ [32] 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

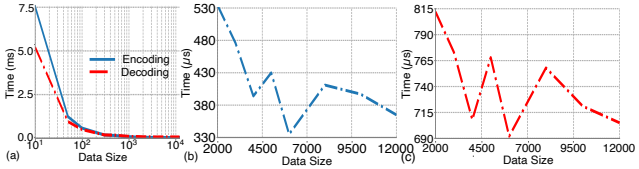


Fig. 2. RaptorQ encoding and decoding time changes with the packet size. (a) Overall trend. (b) Detailed trend of encoding. (c) Detailed trend of decoding

different. Therefore, instead of tracking at packet-level, we should track the sublayer-level reception status. This state is much smaller 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 information to be decoded. 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:

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

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 $0 \le S - v + ku \le kS$ and $u \in \{S, 0\}$. When $v \ge S$, $S - v$ 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.

G. 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 [33]-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.

H. Adapting to Dynamic Channel

As mentioned in Sec. III-E, CSI is required to optimize beamforming. We use the framework developed in ACO [30] and X-array [34] 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 when there is lots of data traffic. 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.

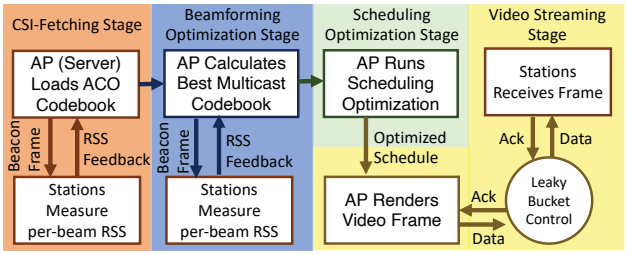


Fig. 3. System Workflow

IV. 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.

A. System Workflow

As shown in Fig. 3, our system begins with fetching CSI using ACO [30]. Then, it calculates the multicast beamweights and measure their respective RSS. The measured RSS of each multicast beamweight are then mapped to throughput of each multicast group. The scheduling optimizer uses these throughputs to calculate the optimized schedule. 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 II). These two commands take $\approx 25\mu$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 reported rate to adjust its sending rate using the leaky bucket-based rate control.

B. 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.

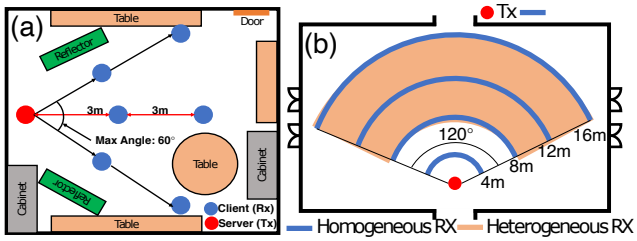


Fig. 4. (a) user placement in testbed experiment. (b) user placement in emulation evaluation

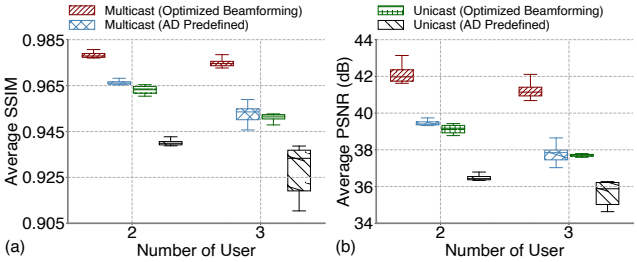


Fig. 5. Testbed result for different number of user. Distance: 3m; MAS: 60°. (a) SSIM. (b) PSNR.

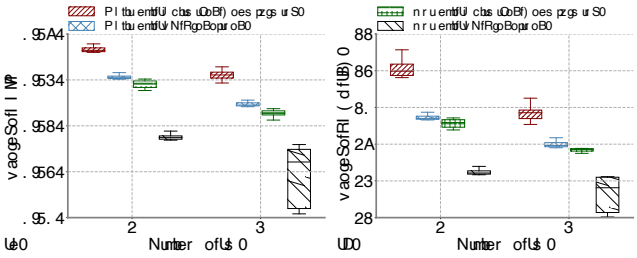


Fig. 6. Testbed result as we vary the distance between server and clients. # Users: 2; MAS: 30°. (a) SSIM. (b) PSNR.

C. Throughput

As shown in Table II, QCA6320 chipset only implements MCS from 0 to 12 excluding 5, 9, and 9.1 in 802.11ad [26]. Existing works [34, 35, 36] use RSS sensitivity table in [26] 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 II, which considers the PHY/MAC overhead. Each value is the average of twenty 10-second *iPerf3* runs.

V. Performance Evaluation

A. 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 data are sent over an emulated link 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. III-C. Each of them is 30FPS and lasts at least 5 minutes.

B. 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.

1) Impact of Beamforming

We compare the following beamforming schemes: (i) optimized multicast beamforming (Sec. III), (ii) pre-defined multicast beam, (iii) optimized unicast beamforming, and (iv) pre-defined unicast beam. They all use the same optimization framework in Section III-D 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 dB, 2.9 dB, and 5.6 dB in 2 users, respectively; 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 out-performs the other schemes by 0.011-0.042

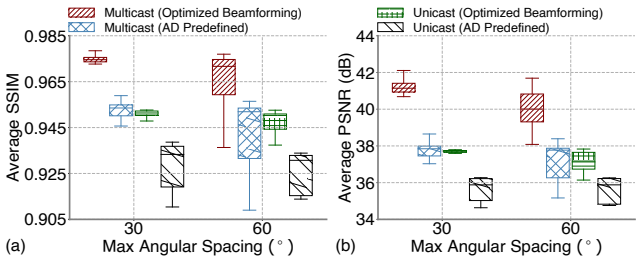


Fig. 7. Testbed comparison of different MAS between the clients. # Users: 2; Distance: 3m. (a) SSIM. (b) PSNR.

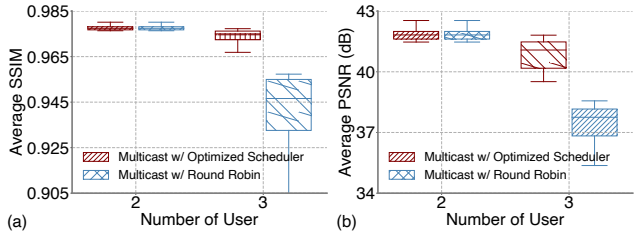


Fig. 8. Testbed comparison between optimized scheduler and round robin. Distance: 3m; MAS: 60°. (a) SSIM. (b) PSNR.

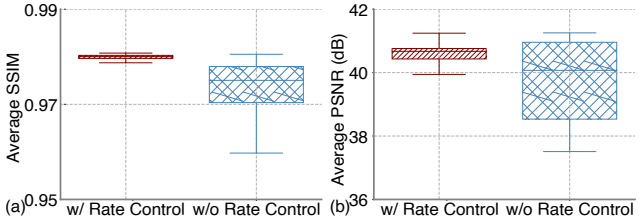


Fig. 9. Testbed comparison between with and without rate control. # Users: 3; Distance: 3m; MAS: 60°; Beamforming: optimized multicast beamforming. (a) SSIM. (b) PSNR.

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 out-performs other alternatives by 0.018-0.048 improvement in SSIM and 3-6 dB improvement in PSNR across all different MAS. Moreover, MAS has little impact on unicast performance but affects multicast performance as we expect.

2) Impact of Schedule

Fig. 8 compares our scheduling with the round-robin [37] 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 out-performs 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.

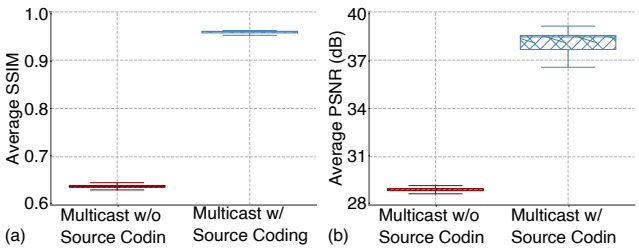


Fig. 10. Testbed comparison between with and without source coding. # Users: 3; Distance: 3m; MAS: 60°; Beamforming: optimized multicast beamforming. (a) SSIM. (b) PSNR.

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. So removing rate control reduces the SSIM by 0.01 and PSNR by 1.3 dB. Moreover, it yields larger variance in video quality since its performance fluctuates with random queue drops. Our leaky bucket-based rate control reduces packet drops at the kernel and maintains high SSIM across all frames over different runs.

4) Impact of Source Coding

Fig. 10 evaluates the impact of source coding. Using source coding out-performs without source coding 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.

C. 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 [38], a popular commercial 3D ray-tracer widely used by the research community [39, 40], 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. Data are sent over an emulated link according to the traces, so no extra packets will be dropped over the channel. In this case, the video quality may be slightly higher than that in testbed under the same experimental setting.

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°.

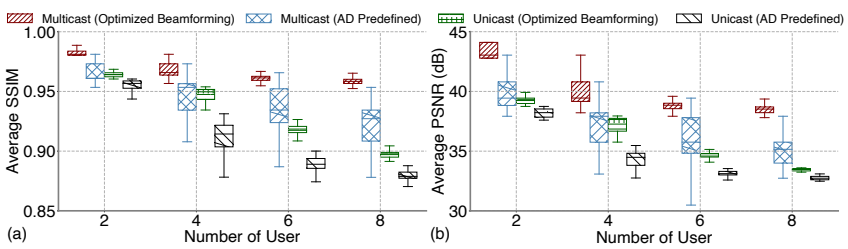


Fig. 11. Emulation comparisons of different # users and beamforming. Distance: 8-16m; MAS: 120°. (a) SSIM. (b) PSNR.

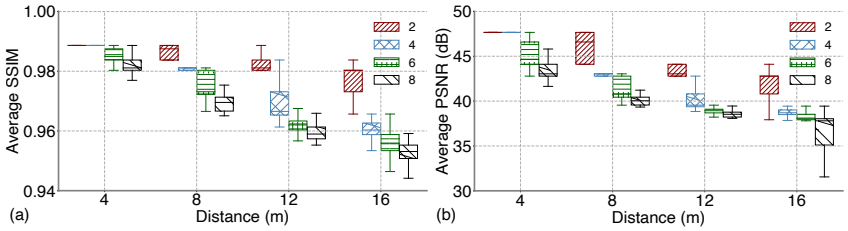
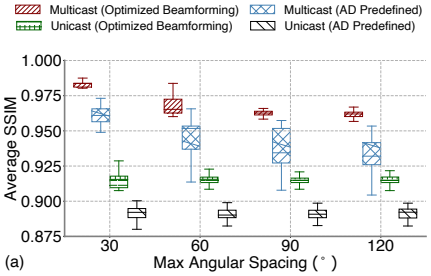


Fig. 12. Emulation comparison of different distances between server and clients. MAS: 120°. (a) SSIM. (b) PSNR.



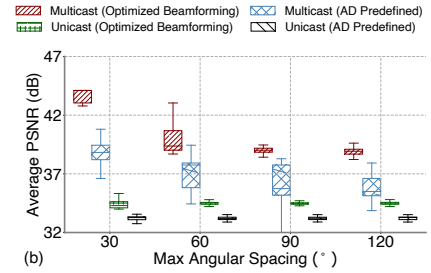


Fig. 13. Emulation comparison of different MAS between clients. # Users: 6; Distance: 12m. (a) SSIM. (b) PSNR.

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, respectively. The corresponding numbers are 0.025, 0.025, and 0.055 in 4 users, 0.03, 0.04, and 0.075 in 6 users, 0.035, 0.06, and 0.083 in 8 users, respectively. The corresponding PSNR advantage ranges from 2.6 dB - 4.7 dB in 2 users scenario, 3 dB-5 dB in 4 users scenario, 3 dB-5.8 dB in 6 users scenario, and 3.3 dB-5.9 dB in 8 users scenario.

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 as the distance and the number of users vary. 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. In all cases, optimized or pre-defined multicast beamforming out-performs 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.

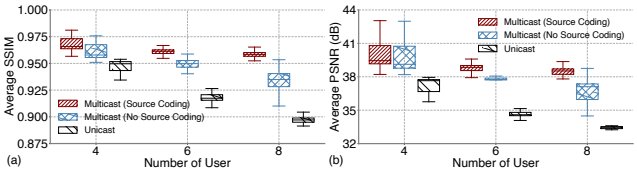


Fig. 14. Emulation comparisons of with and without source coding. Distance: 8-16m; MAS: 120°. (a) SSIM. (b) PSNR

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 video quality by around 0.005-0.025 SSIM and 1-3dB PSNR. The emulation result draws the same conclusion as our testbed evaluation.

3) Impact of Scheduling

Fig. 15 compares our scheduling with round-robin scheduling while both use optimized multicast beamforming. Similar to Fig. 8, when there are 2 users, there is only one multicast

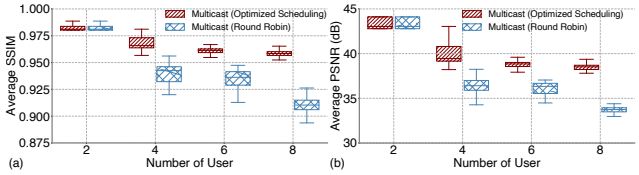


Fig. 15. Emulation comparisons of different scheduling algorithms. Distance: 8-16m; MAS: 120°. (a) SSIM. (b) PSNR

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 and 2.5, 2.7, and 4.8dB in PSNR under 4, 6, and 8 users scenarios, respectively. As expected, the importance of scheduling increases with the number of users.

4) Trace-driven Mobile Experiment

As explained in Sec.III-H, 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 [26], 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 weights, time allocation and packet schedule according to the each measurement in the 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. [10] implements 7 ABR algorithms. [41] reports Robust MPC [42] and Fast MPC are two of the best ABR algorithms for live video streaming, which are used 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 a 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. Robust MPC and Fast MPC differ from Real-time Update by using H264 as the video codec, only applying our rate control approach, and adopting round-robin scheduling for multicast streaming.

For brevity, we only show SSIM in this section. 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.

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 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 consistently above 0.95. 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

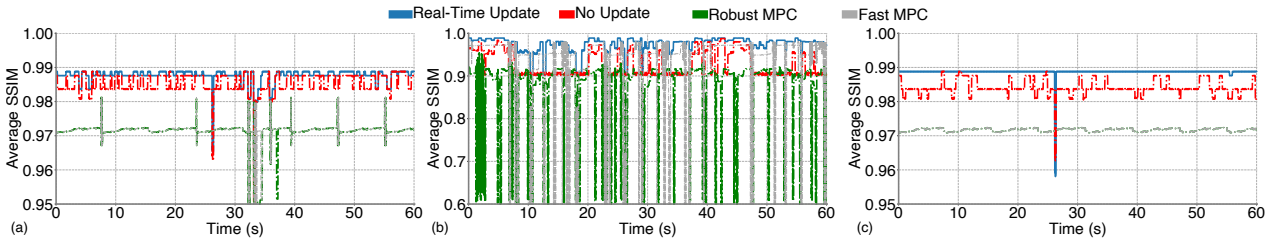


Fig. 16. Emulation result of serving 1 receiver when (a) receiver is moving under High RSS ($\geq -61dBm$), (b) receivers are moving under Low RSS ($< -61dBm$), and (c) environment is moving.

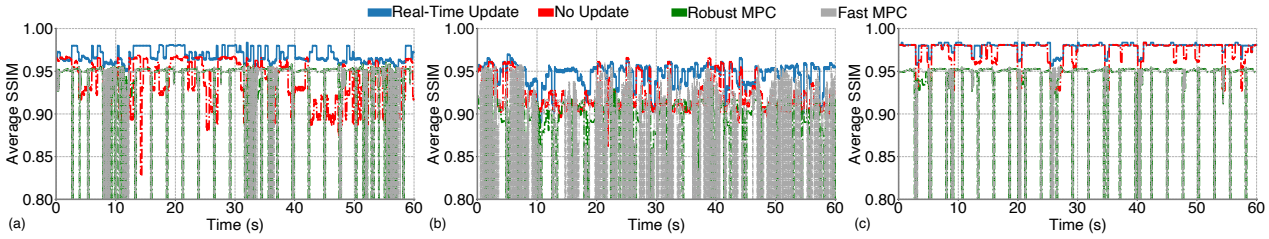


Fig. 17. Emulation result of serving 3 receivers when (a) two receivers are moving under High RSS ($\geq -61dBm$), (b) two receivers are moving under Low RSS ($< -61dBm$), and (c) environment is moving.

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 people move, the network condition fluctuates. We see that the No Update is not affected much because 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.

VI. Conclusion

We develop 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 develop an effective end-to-end system. Our extensive experiments demonstrate its effectiveness.

Acknowledgement

This work is supported in part by NSF Grant CNS-2212297. We appreciate the insightful feedback from ICDCS 2024 anonymous reviewers.

References

[1] J. Archer, "4k tvs: 9 reasons you should buy one – and 9 more why you shouldn't," 2015. [Online]. Available: https://www.forbes.com/sites/johnarcher/2014/11/17/4k-tvs-9-reasons-you-should-buy-one-and-9-more-why-you-shouldnt/

[2] "Limelight," 2022. [Online]. Available: https://www.limelight.com/

[3] Edge, "How edge compute is enabling better and more affordable gaming experiences," 2019. [Online]. Available: https://www.section.io/blog/edge-computing-gaming-benefits

[4] Y. Song, C. Ge, L. Qiu, and Y. Zhang, "2ACE: Spectral Profile-driven Multi-resolutional Compressive Sensing for mmWave Channel Estimation," in *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MobiHoc '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 41–50.

[5] T. Stockhammer, "Dynamic adaptive streaming over HTTP –: standards and design principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 133–144.

[6] R. Zhao, T. Woodford, T. Wei, K. Qian, and X. Zhang, "M-Cube: a millimeter-wave massive MIMO software radio," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New York, NY, USA: Association for Computing Machinery, 2020.

[7] G. Baig, J. He, M. A. Qureshi, L. Qiu, G. Chen, P. Chen, and Y. Hu, "Jigsaw: Robust Live 4K Video Streaming," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019.

[8] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[9] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[10] 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 *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 610–625.

[11] 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.

[12] 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*, ser. CRAB '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 61–68.

[13] H.-R. Shao, J. Hsu, C. Ngo, and C. Kweon, "Progressive transmission of uncompressed video over mmw wireless," in *2010 7th IEEE Consumer Communications and Networking Conference*, 2010, pp. 1–5.

[14] 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*, vol. 46, no. 12, 2008.

[15] X. Li, S. Paul, and M. Ammar, "Layered video multicast with retrans-

missions (lvmr): Evaluation of hierarchical rate control," in *Proceedings of the IEEE INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, vol. 3. IEEE, 1998, pp. 1062–1072.

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

[17] B. J. Vickers, C. Albuquerque, and T. Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 720–733, 2000.

[18] C. Guo, Y. Cui, D. W. K. Ng, and Z. Liu, "Multi-quality multicast beamforming with scalable video coding," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5662–5677, 2018.

[19] R. Kuschnig, I. Kofler, and H. Hellwagner, "An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H.264/SVC," in *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, ser. MMSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 157–168.

[20] 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, Sept 2007.

[21] S. Naribole and E. Knightly, "Scalable Multicast in Highly-Directional 60-GHz WLANs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2844–2857, 2017.

[22] M. Chitnis, P. Pagano, G. Lipari, and Y. Liang, "A Survey on Bandwidth Resource Allocation and Scheduling in Wireless Sensor Networks," in *2009 International Conference on Network-Based Information Systems*, 2009, pp. 121–128.

[23] I. Sousa, M. P. Queluz, and A. Rodrigues, "A survey on QoE-oriented wireless resources scheduling," *Journal of Network and Computer Applications*, vol. 158, p. 102594, 2020.

[24] XIPH, "Video dataset," 2021. [Online]. Available: https://media.xiph.org/video/derf/

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

[26] "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 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016.

[27] "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 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[28] "Qualcomm QCA9500 Chipset," 2017. [Online]. Available: https://www.qualcomm.com/products/qca9500

[29] S. Sur, I. Pefkianakis, X. Zhang, and K.-H. Kim, "Practical MU-MIMO user selection on 802.11ac commodity networks," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 122–134.

[30] J. Palacios, D. Steinmetzer, A. Loch, M. Hollick, and J. Widmer, "Adaptive Codebook Optimization for Beam Training on Off-the-Shelf IEEE 802.11ad Devices," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 241–255.

[31] M. J. Lopez, "Multiplexing, scheduling, and multicasting strategies for antenna arrays in wireless networks," PhD thesis, MASSACHUSETTS INST OF TECHNOLOGY, Boston, MA, 2002.

[32] C. Berner, "Rust-based Implementation of RaptorQ," 2021. [Online]. Available: https://github.com/cberner/raptorq

[33] M. Butto, E. Cavallero, and A. Tonietti, "Effectiveness of the 'leaky bucket' policing mechanism in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 335–342, 1991.

[34] 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 *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New York, NY, USA: Association for Computing Machinery, 2020.

[35] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "BeamSpy: Enabling Robust 60 GHz Links Under Blockage," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 193–206.

[36] S. Wang, J. Huang, and X. Zhang, "Demystifying millimeter-wave V2X: towards robust and efficient directional connectivity under high mobility," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New York, NY, USA: Association for Computing Machinery, 2020.

[37] R. V. Rasmussen and M. A. Trick, "Round Robin Scheduling–A survey," *European Journal of Operational Research*, vol. 188, no. 3, pp. 617–636, 2008.

[38] "Wireless Insite 3D Wireless Prediction Software," 2020. [Online]. Available: https://www.remcom.com/wireless-insite-em-propagation-software

[39] Z. Jiang, S. Zhou, Z. Niu, and C. Yu, "A Unified Sampling and Scheduling Approach for Status Update in Multiaccess Wireless Networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 208–216.

[40] K. Joshi, D. Bharadia, M. Kotaru, and S. Katti, "WiDeo: Fine-grained Device-free Motion Tracing using RF Backscatter," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 189–204.

[41] 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 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2642–2646.

[42] 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*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 325–338.