Supporting Untethered Multi-user VR over Enterprise Wi-Fi

Xing Liu[1] Christina Vlachou[2] Feng Qian[1] Kyu-Han Kim[2]
[1]University of Minnesota – Twin Cities, USA [2]Hewlett Packard Labs, Palo Alto, USA

ABSTRACT

In this positioning paper, we propose Chord, a holistic multi-user VR system for untethered mobile devices over 802.11ac/ax Wi-Fi. Taking a cross-layer approach, Chord brings numerous innovations to the application-layer design and VR-aware wireless network optimizations. We present our design and its preliminary evaluation on commodity smartphones and 802.11ac Wi-Fi, to demonstrate the feasibility of Chord.

CCS CONCEPTS

• **Networks → Wireless local area networks**; • **Computing methodologies → Virtual reality**.

KEYWORDS

Virtual Reality; Multi-user VR; Enterprise Wi-Fi.

ACM Reference Format:
Xing Liu, Christina Vlacho, Feng Qian, and Kyu-Han Kim. 2019. Supporting Untethered Multi-user VR over Enterprise Wi-Fi. In *29th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '19), June 21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3304112.3325605

1 INTRODUCTION

VR is becoming important in enterprise environments with many applications in education, training, collaborative meetings, and product design. Enterprises are embracing VR to reduce costs (production, travel, maintenance, training) and to enhance work productivity. This trend is well supported by a plethora of VR hardware, such as VIVE BE [4] and Oculus Go for business [6], as well as by VR software platforms for collaborative projects [5] or meetings [3].

It is well known that tethered head-mounted VR gears are not safe and do not support user mobility. The multi-user VR scenario makes these disadvantages even worse. Therefore, in this paper, we propose a system, called Chord, that supports untethered multi-user VR on commodity mobile devices, such as smartphones and tablets, without requiring expensive, specialized VR hardware. Chord needs only one-time installation of a thin client-side app on mobile devices. Then, at runtime, a remote (or edge) server distributes VR content to the client devices based on users' viewports and interaction over enterprise Wi-Fi networks.

Despite recent efforts on VR research, providing high-quality, immersive multi-user experience still poses numerous challenges. While some of them, such as limited battery life and computation power on mobile devices, have been considered for single-user VR [9, 12, 15], little research has been conducted on multi-user VR, which brings a new set of *scalability* and *diversity* challenges. How to support tens of untethered users interacting simultaneously over the state-of-the-art wireless infrastructure [7, 15]? How to make the server scale when executing complex scheduling and rendering operations? How to accommodate user devices with heterogeneous hardware configurations? The last one is an important and practical requirement, given that "BYOD" (bring-your-own-device) policies are becoming increasingly popular in enterprises [2].

By taking a cross-layer, interdisciplinary approach, Chord brings numerous innovations to both the application-layer design and VR-aware wireless network optimizations.

The Application-layer Design (§3.1). Chord takes several unique approaches to ensure better scalability while maintaining high content quality and friendliness to diverse VR client devices. *(i)* By default, the server performs offline pre-rendering of all possible scenes. This helps dramatically reduce the client-side overhead and make the server free of expensive rendering operations at runtime. *(ii)* Chord differentiates interactive objects from non-interactive objects on which users have different levels of expectations on Quality-of-Experience (QoE). For objects in each category, Chord provides both the (default) rendered format and the unrendered format. Such a dual-representation design is a key to accommodating the heterogeneity of client devices. *(iii)* To ensure the limited network resources can accommodate as many clients as possible, Chord allows the content quality to be dynamically adjusted. This is achieved through a centralized Adaptive Content Quality Control (AQC) module that jointly considers the clients' wireless channel capacity, their rendering capability, users' preference, and the QoE model. AQC also interacts with the lower wireless layer to obtain accurate bandwidth estimation and to offer hints on QoE-aware resource allocation to the lower layer. *(iv)* Chord intelligently streams the VR content based on predicting the user's 6-DoF viewport movement, leading to even more efficient network usage.

The Wireless Network Optimization (§3.2). Chord builds on the state-of-the-art 802.11ac and the next-generation 802.11ax standards. Harnessing a cross-layer design, the optimizations we propose are the following. (*i*) Chord employs the Wi-Fi lower-layer information to facilitate accurate throughput estimation that benefits AQC's decision making. (*ii*) We quantitatively study the option of performing multicast Wi-Fi transmission and decide that it is not a good fit for multiuser VR. (*iii*) Chord users exhibit constant and various levels of mobility, from moving the head and the device, to walking in a room. We thus perform judicious optimizations for MU-MIMO grouping, scheduling, and rate control to make them VR-aware. (*iv*) We also introduce cross-layer mechanisms to reduce the uplink delay associated with reporting users' viewport and interactions. The above components are typically executed at the Wi-Fi AP and will be seamlessly integrated into the Chord system.

NOSSDAV '19, June 21, 2019, Amherst, MA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6298-6/19/06…$15.00
https://doi.org/10.1145/3304112.3325605

To our best knowledge, this positioning paper presents a first research proposal of developing a holistic multi-user VR system for untethered COTS mobile devices, by incorporating intra-disciplinary ideas from cross-layer system design, wireless networks, visualization and graphics, theoretical modeling and optimization, and machine learning.

The remainder of this paper is organized as follows. After presenting the background and related work in §2, we describe the design of Chord in §3. We then present the preliminary results in §4 to further motivate and demonstrate the feasibility of Chord, before concluding the paper in §5.

2 BACKGROUND AND RELATED WORK

Untethered VR over Wireless Links. Several previous studies have used mmWave links for VR [7, 18]. Typically, mmWave can support transmitting uncompressed VR frames for a single user, which eliminates decompression latencies at the clients. However, mmWave today has several limitations: (*i*) a mobile client's CPU and battery may not be able to support such fast packet processing [12]; (*ii*) being directional, mmWave suffers from blockage (in particular in a multi-user setting) and can lead to frame losses or bursty errors when beam tracking algorithms are not efficient [7]; and (*iii*) most commodity phones today do not yet support mmWave.

An alternative approach to mmWave is to deliver VR content via omni-directional radio such as LTE [15] and traditional Wi-Fi [12], or to save the entire scene locally [9]. In these cases, the large sizes of uncompressed VR scenes make directly storing or streaming VR content over onmi-directional radio infeasible. Therefore, a practical VR system has to leverage compression, which can reduce these sizes by 90% on average at the cost of a few millisecond decoding latency per frame [12]. Our system also judiciously uses compression, yet in addition employs various optimizations to address the unique challenges brought by multi-user VR.

802.11ac/ax: Wi-Fi is dominant in enterprise. The state-of-the-art 802.11ac standard supports multi-gigabit speed (up to 6.9 Gbps [1] in theory), making it a promising wireless technology to deliver VR content for multiple concurrent users. In practice, typical mobile phones support up to 866 Mbps, a rate sufficient for compressed VR frames, as shown by recent studies [12]. In order to determine the appropriate PHY rate per station, an access point (AP) employs the *rate control* algorithm to compute the packet error-rate (PER) based on frame losses and sub-frame error-rate (SFER). PER can increase, due to either bad channel conditions or packet collisions with other stations. 802.11ac also supports multiuser (MU)-MIMO, which allows concurrent downlink data streams from an AP to a group of clients. Thanks to its concurrency, MU-MIMO yields significant gains compared to single-user (SU) transmissions. However, if there exists high inter-user interference, the gains will decrease, and clients will suffer from high SFER. MU-MIMO algorithms rely on PER to estimate inter-user interference and to form groups of users with uncorrelated channels [14].

802.11ax is the latest Wi-Fi standard that promises 11 Gbps theoretical rates and further enhances multi-user performance by using AP-centric OFDMA (Orthogonal Frequency Division Multiple Access) [10], where the Wi-Fi bandwidth can be effectively shared among users with simultaneous uplink and downlink transmissions. Contrary to 802.11ac, 802.11ax does not use CSMA/CA to share the channel but centralizes both uplink and downlink OFDMA decisions at the AP. OFDMA can significantly reduce delays and minimize collisions, hence making 802.11ax even more promising for multi-user VR.

3 THE DESIGN OF CHORD

Chord scales multi-user VR in enterprise environments. As shown in Figure 1, our design has a cross-layer nature by optimizing both the application layer (§3.1) and the wireless layer (§3.2), and letting them be aware of and properly interact with each other, in order to ensure good scalability while maintaining high content quality.

3.1 Application-Layer Design

Server-Side Pre-Rendering. A key design decision we need to make for Chord is how to represent and store VR content at the server side. We consider three possible approaches. (*i*) The server stores the content as 3D models, which will be transferred to and rendered by the client. However, prior studies [12] and our preliminary results (§4) indicate that today's commodity mobile devices are not powerful enough to perform fully local rendering for high-quality VR. (*ii*) The server performs rendering in real-time and sends the rendered results as compressed video frames to clients. Doing so alleviates the client-side overhead but makes the server not scalable at all. This is in particular because of the high video encoding overhead. We conduct an experiment on a workstation with an Nvidia GTX 1080 GPU. The workstation can only achieve encoding performance of 92 FPS, 199 FPS, and 342 FPS for 4K, 2K, and 1080p resolution, respectively, which clearly cannot support many clients. (*iii*) The server performs one-time offline rendering for *all* possible viewports, and caches the rendered and encoded frames in its storage. At the runtime, the server simply acts like a file server by transmitting the pre-generated video frames according to clients' reported viewport position and orientation. This approach dramatically reduces the server's runtime workload at the cost of high disk space usage, which is typically not a big issue given the cheap storage today.

Given its advantages, Chord adopts the third option as the default "base solution" upon which numerous optimizations will be introduced. Specifically, in the offline rendering phase, Chord enumerates all possible positions in the scene. For each position, Chord renders its 360° view into a panoramic video frame. In this way, the whole (discrete) 6-DoF space is fully exercised. The frames are then encoded into a video stream whose individual GOPs (group of pictures) can be fetched by the client. Note that a similar approach was taken by FlashBack [9], which caches everything on the local device as opposed to on the server as in Chord.

Dual Content Representations. Chord treats interactive (IA) and non-interactive (NIA) objects differently. IA objects are those users can interact with, and are typically situated in the foreground of a scene. Their examples include the target product in a promotion or virtual customers whom users can interact with. In contrast, NIA objects are non-interactive and are typically situated in the background. In most VR scenes, the static background, such as an office room, can be treated as a single "logical" NIA object. IA objects are more important than NIA objects, and should be rendered with a higher quality and lower latency if possible.

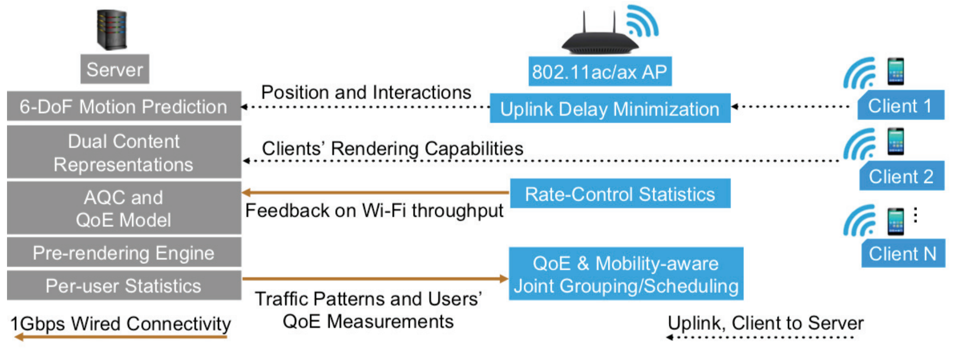

Server
6-DoF Motion Prediction
Dual Content Representations
AQC and QoE Model
Pre-rendering Engine
Per-user Statistics
1Gbps Wired Connectivity
Position and Interactions
Clients' Rendering Capabilities
Feedback on Wi-Fi throughput
Traffic Patterns and Users' QoE Measurements
802.11ac/ax AP
Uplink Delay Minimization
Rate-Control Statistics
QoE & Mobility-aware Joint Grouping/Scheduling
Uplink, Client to Server
Client 1
Client 2
Client N

Figure 1: The Chord **Architecture. The downlink data paths from AP to clients are not shown.**

In Chord, the server prepares two representations of each IA object: the *rendered* and *unrendered* format, which incur an important tradeoff. The rendered format is a video stream, with each frame rendering the object from a particular view, generated using the above "exhaustive enumeration" approach[1]. It is typically easy to be processed by the client, which, however, needs to continuously fetch the frames as the viewport changes or an interaction occurs, incurring additional network delay. The client can alternatively cache *all* frames of an IA object, but that requires a substantial amount of bandwidth and local storage. On the other hand, the unrendered format contains raw 3D meshes and textures. Unless the model is simple, it requires more processing capability from the client. However, it offers a key benefit of its compact representation that consumes much less bandwidth and takes much less space to cache compared to the rendered format. Unrendered models provide other benefits such as allowing the client to perform custom rendering (e.g., highlighting the surface of a model).

Providing dual representations of IA objects offers great flexibilities to client devices with heterogeneous hardware configurations. For example, low-end and high-end devices may choose to use the rendered and the unrendered format, respectively. The two formats can also be used in a hybrid manner. For example, when there is spare bandwidth, a client can opportunistically prefetch unrendered models and cache them, leading to reduced bandwidth usage in the long term. Note that the server can also prepare dual versions for NIA objects, but a typical mobile client may not be able to render both IA and NIA objects due to its limited 3D rendering capability. We will develop an algorithm that adaptively selects the appropriate format based on several factors including the device's rendering capability, the available bandwidth, and the cache size. The algorithm will be integrated into the AQC module described below.

Adaptive Content Quality Control (AQC) is another mechanism in Chord to facilitate better scalability from the network's perspective. Its basic concept is borrowed from the rate adaptation mechanism of video streaming, where the client player dynamically changes the video quality based on its estimation of the network bandwidth. However, AQC in Chord differs from traditional rate adaptation in several key aspects. (*i*) AQC is performed on the server side, which has a visibility of *all* clients and therefore can run a centralized scheduling algorithm to maximize the global QoE. (*ii*) In a typical enterprise deployment scenario, the server and the AP are within the same administrative boundary. This creates rich cross-layer optimization opportunities [16]. On one hand, the server can leverage the lower-layer information at the AP such as accurate channel capacity estimation to better assist the app-layer AQC. On the other hand, based on the app-layer requirements, the server can pro-actively hint the lower-layer scheduling and resource allocation decisions, as to be detailed in §3.2. (*iii*) Recall that Chord offers two types of representations: rendered frames and unrendered models. Chord therefore needs to create separate quality versions and run separate AQC algorithms for them. We plan to thoroughly investigate both. In particular, for unrendered 3D models, very few studies have been done on its AQC or on the underlying QoE metrics that dictate AQC's decisions. (*iv*) AQC will also leverage the 6-DoF motion prediction (to be detailed shortly) to aid its scheduling decisions.

Overall, we plan to integrate the above four design points into a holistic AQC framework which, in a cross-layer manner, adaptively selects for each client: (*i*) the format for each object, (*ii*) the quality level for each IA object, and (*iii*) the quality level for each NIA object. The framework will take several factors into consideration: the clients' wireless channel capacities, their rendering capabilities (which clients will inform the server), users' preferences (e.g., the weight of IA vs. NIA objects), and the QoE model. We will derive an optimization-based formulation for online scheduling. Here a potential challenge is to make the actual solving process efficient and scalable w.r.t. the number of clients and the number of objects. We plan to tackle it by, for example, developing robust heuristics and grouping similar users/objects.

6-DoF Motion Prediction. In Chord, VR content is continuously streamed to each client according to its viewport. To hide the end-to-end "motion-to-photon" delay, the content needs to be properly prefetched. A simple approach taken by prior work [12] is to blindly prefetch the panoramic views at the locations that are adjacent to a user's current position. Chord instead brings in more intelligence by performing 6-DoF motion prediction. For example, if the user is moving towards a particular direction, then the rendered frames or unrendered models in that direction will be more aggressively prefetched, whereas the content in other directions may be fetched at a lower quality or even be skipped to save the

[1]For an IA object, the enumeration needs to exercise at least 7 dimensions: 3D relative position, 3D object rotation, and the animation sequences of one or more dimensions. Each frame will be rendered with depth information so that multiple objects can be properly combined on the client side.

bandwidth. Prior studies indicate that for 360° panoramic streaming, viewers' head movement (in 3-DoF) exhibits good short-term predictability using lightweight machine learning algorithms [11, 13]. We thus plan to investigate the 6-DoF case. Here a key hypothesis to be validated through user studies is that the viewport's angular movement (yaw, pitch, roll) and the translational movement (the 3D position) can first be predicted separately, and then their results can be combined to obtain the final 6-DoF prediction. In this way the overall complexity of the prediction will be significantly reduced.

In addition, since the server has the knowledge of all users' viewport movement trajectories, the server can perform (online or offline) *crowdsourcing* analysis to compute the "hotspots" that users are commonly interested in, in order to further assist viewport prediction. We expect that the content of enterprise VR applications, such as educational materials and product demonstration, usually has hotspots that are easily identifiable through crowdsourcing and/or content analysis. We will also validate this through IRB-approved user studies.

3.2 Network-Level Optimizations

Orchestrated by the modules described in §3.1, Chord's network-level optimizations seek to save bandwidth, increase throughput, and reduce delay. We make a reasonable assumption that the Chord server can directly manipulate configurations of the Wi-Fi AP that is dedicated to the Chord system.

Accurate Throughput Estimation. Chord leverages the wireless lower-layer information to facilitate accurate throughput estimation, which benefits AQC's decision making. Specifically, the Wi-Fi AP gathers statistics in the rate control module in firmware. The statistics consist of the average PHY rate ϕ_i, the average subframe error rate (SFER) σ_i, and the average size of SU/MU-MIMO groups g_i per client i. Then we can compute the link's maximum achievable throughput as $\phi_i(1-\sigma_i)(1-\text{TX}_{overheads})$.[2] In addition, we need to account for the fact that (in theory) all N stations share equitably the medium, and also that the network has some gains when MU-MIMO groups are formed. Hence, we estimate the final throughput of client i as $P_i = g_i\phi_i(1-\sigma_i)(1-\text{TX}_{overheads})/N$, by leveraging the information exposed by the rate control and the joint grouping-scheduling modules of the AP. Our current AP-based technique has 1–5% relative estimation error for 802.11ac based on our lab tests (evaluation details omitted).

We further propose to enhance the above approach with traditional app/transport-layer throughput measurement, to make the throughput estimation even more accurate. The rationale is that the source of errors of the two approaches differs: for AP-based measurement, the airtime might not be always shared equitably among the stations due to interference or scheduling unfairness; the app/transport-layer measurement, denoted as T_i^{app} for client i, might be bounded by the amount of traffic sent to the client, which can be lower than the actual capacity. Hence, these two techniques can be combined for increased accuracy: if T_i^{app} and P_i deviate from each other, then we examine the packet loss rate (denoted as l_i): If $l_i \approx 0\%$, then we are below the capacity and the AP's estimation applies. Otherwise, we reach or are close to the capacity so the throughput is estimated as T_i^{app}.

[2]$\text{TX}_{overheads}$ is mainly due to IP/UDP headers and the Wi-Fi MAC layer and it is measured to be 30–40%.

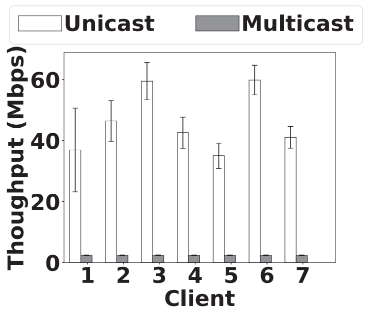


Figure 2: 802.11ac downlink unicast vs. multicast.

No Multicast Adoption. In a multi-user setting, a natural idea one may think of is to perform multicast. Indeed, it was recently proposed for multi-user 360° video streaming [8]. We argue, however, that multicast is not suitable for Chord from the network's perspective.

• *Why is multicast not suitable for commodity high-rate Wi-Fi?* Wi-Fi networks have adaptive rate control. Depending on the user's channel quality and location (distance from AP, objects in between, mobility, etc.), every user has its own optimal/maximum rate at which she can decode data. Hence, multicast/broadcast transmissions have to ensure all involved users can reliably decode packets, as these transmissions do not have link-layer acknowledgements by default. In commodity APs, multicast transmissions are handled at the most robust rate (e.g., 6Mbps) like other crucial management messages such as beacons. Figure 2 demonstrates this feature through an experiment of performing multicast to 7 clients in a room from an off-the-shelf 802.11ac AP. As shown, the per-client multicast throughput is much lower than the per-client throughput of performing concurrent unicast to the same set of clients. To achieve higher rates, one has to introduce rate control and acknowledgements for multicast transmissions over different client groups. This brings more complexity and is not a scalable solution.

Optimizing MU-MIMO Grouping and Scheduling. 802.11ac/ax joint MU-MIMO user grouping and scheduling is crucial for multi-user applications. The grouping protocol may introduce high delays and low throughput if it selects the wrong users to group in an MU-MIMO transmission, as users with correlated channels cause high packet losses (SFER) due to interference (§2). For scheduling, ideally the protocol has to be short-term fair to ensure low access delay for all users; the protocol should also determine the optimal frame duration based on the traffic dynamics.

In Chord, we propose to group users by minimizing interference and to avoid delays by crafting VR-aware frame aggregation. Specifically, we propose two optimizations. First, we can blacklist groups with SFER yielding prohibitive delays whose threshold is dictated by the Chord server based on the VR QoE requirements. Second, the Chord server can also inform the AP about the size distribution of the "atomic" data blocks to be transmitted, such as GOPs of rendered frames. The AP can then use this information to make strategic frame aggregation decisions, such as limiting the transmission duration, to maintain the proper scheduling granularity without delaying other users.

Uplink Delay Reduction. In Chord, a client needs to continuously transmit uplink (UL) meta-data that contains the user's position, orientation, and controller inputs. Due to the heavy downlink (DL) traffic that causes CSMA/CA deferrals or collisions, users might experience tens of ms delay for meta-data transmissions over 802.11ac. 802.11ax Wi-Fi can also incur delays, as users need to send buffer status reports (BSR) to earn OFDMA UL slots from the AP. To this end, we propose three AP-based solutions to reduce the UL delay while maintaining good DL performance without modifying the client-side Wi-Fi stack.

- *802.11ac AP silent slots.* Typically users' meta-data is only a few tens or hundred bytes, hence its transmission should complete in a very short interval. When the queue length of DL traffic is short, the AP can add short delay or silent slots to facilitate UL transmission. This reduces the UL delay caused by CSMA/CA deferrals or its incurred collisions. The Chord server can provide hints on how to set the frequency and duration of such silent slots.
- *IEEE 802.11ax VR-aware UL OFDMA scheduling.* In the next-generation OFDMA, transmissions are completely controlled by the AP, which, when collaborating with Chord, can schedule periodically all clients with small uplink meta-data on a single frame transmission. In this case, clients are completely in sync and transmitting meta-data simultaneously without any collisions (contrary to 802.11ac). The Chord server will inform the AP about the UL resource allocation decisions, which depend on the bandwidth, the number of clients, the meta-data size, and the users' viewport movement dynamics. The AP can then proactively assign UL slots without requiring BSR.
- *Uplink transmissions over a side channel.* Another possible solution would be to schedule uplink meta-data over another medium such as Bluetooth. In theory, Bluetooth can support the data rate demand of VR meta-data and multiple concurrent connections to a master (the Chord server).

4 PRELIMINARY EVALUATION

We present three pieces of preliminary results. (*i*) The server-side pre-rendering approach is highly beneficial and can be well supported by state-of-the-art Wi-Fi. (*ii*) COTS mobile devices indeed exhibit high heterogeneity and thus need differential treatment as proposed in §3.1. (*iii*) Vanilla 802.11ac interacts poorly with Chord in particular when many concurrent users are present. This motivates our network-layer optimizations proposed in §3.2.

Server-Side Pre-Rendering vs. Local Rendering. We compare the performance of rendering a VR scene locally on a smartphone with that of using the server-side pre-rendering approach, to motivate the latter. Specifically, we pick a subset of the *Viking Village* scene offered by the Unity Asset Store. It consists of 1,482 models that are non-interactive (NIA). We use a Samsung Galaxy Note 8 (SGN8) as the client device. For local rendering, we directly utilize the Unity API for Android to render the models at the 1440p resolution. For our pre-rendering approach, we discretize the entire scene (100×100 m^2) into a grid of 5×5 cm^2 blocks. We employ a commodity server to render a panoramic frame offline at each grid location. We then encode the frames into an H.264 video stream where each GOP (group-of-picture) consists of 9 frames within each 3×3 grid, under three quality levels defined by the Constant Rate Factor (CRF): CRF=23 (high quality), 29 (medium quality), and 35 (low quality). We set the encoded resolution to be also 1440p. The total size of the entire pre-rendered scene is 288GB when CRF=23. At the run time when a user is navigating in the scene, the client fetches its nearby frames from the server, and then decodes, caches, and renders the frames when needed. On the client side, we use the low-level Android *MediaCodec* API for decoding, and develop a cache with a capacity of 100 decoded frames implemented using the OpenGL Frame Buffer Object with simple FIFO cache replacement.

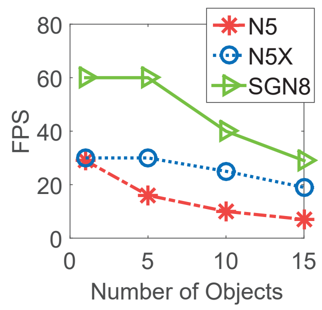
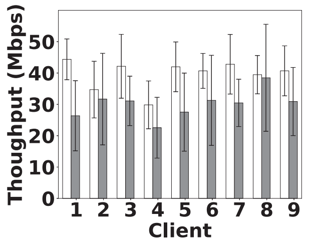


Figure 3: Rendering performance of 3 smartphones.

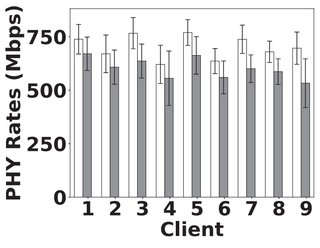
We now compare the rendering performance of the above two approaches. For local rendering, we observe a poor performance of only 5FPS. In contrast, for the pre-rendering approach, assuming the bandwidth is not the bottleneck, we can achieve an FPS of 135 on SGN8 at 1440p – a 26× improvement. We also measure the bandwidth consumption for the pre-rendering approach. We first consider an ideal case where all fetched GOPs are consumed by the viewer (i.e., having perfect viewport movement prediction). The per-user throughput is measured to be 40Mbps, 15Mbps, and 7Mbps for CRF=23, 29, and 35, respectively. We then consider a pessimistic case where only 30% of fetched GOPs are consumed (very poor viewport prediction). The throughput is measured to be 132Mbps, 50Mbps, and 22Mbps, respectively, for the three quality levels. Overall, we believe the bandwidth requirement is reasonable given the high capacity of 802.11ac/ax. The results also highlight the importance of AQC and viewport prediction in Chord.

The Heterogeneity of Commodity Mobile Devices is a key aspect that Chord considers. To demonstrate that, we consider three COTS smartphones: Google Nexus 5 (released in 2013), Google Nexus 5X (2015), and SGN8 (2017). They correspond to a low-end, medium-end, and high-end device, respectively, as the time when this paper was written. We use them to render several complex "Barbarian Warrior" objects obtained from the Unity Asset Store (http://bit.ly/2Ozlshk). The object belongs to the interactive (IA) category, with its interaction sequence pre-defined here for simplifying our experiments. It consists of 14,075 vertices and 18,062 triangles. We utilize the Unity API for Android to render the objects locally on the client. Figure 3 shows the rendering performance in FPS by varying the number of objects on the three devices. The results show highly diverse rendering performance that motivates our dual representation design – where clients can flexibly choose the content format, and AQC – where clients can fetch lower-quality objects for faster rendering.

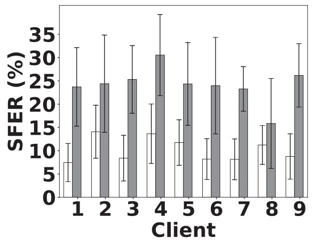
802.11ac MU-MIMO Performance. We experimentally investigate the multi-user performance of 802.11ac in an emulated VR



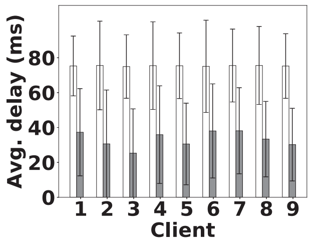
(a) Throughput



(b) PHY rates



(c) SFER



(d) Scheduling delay

Figure 4: 802.11ac multi-user SU/MU-MIMO downlink performance with 9 clients.

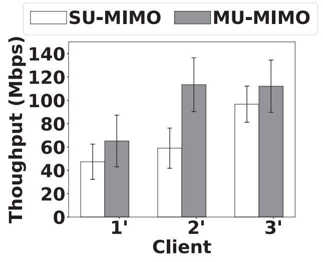


Figure 5: 802.11ac multi-user SU/MU-MIMO downlink performance with 3 clients.

context. We modify the firmware of an enterprise-grade AP to collect statistics and to use either MU- or SU-MIMO. We first demonstrate that MU-MIMO can yield significant gains if properly configured for a small number of users. In our setup, we distribute three clients in a large lab emulating a VR room. The clients have un-correlated channels, as verified by the channel state information. They are first grouped with MU-MIMO and then using SU-MIMO to concurrently download data from a local server. Figure 5 uncovers that MU-MIMO can yield 44% throughput gains compared to SU-MIMO (290 vs. 201Mbps total throughput).

We then try to scale up the above setup by increasing the number of clients from 3 to 9 and repeating the same concurrent data download experiment. In this case, we observe the opposite results: SU-MIMO outperforms MU-MIMO in terms of throughput, PHY rates, and SFER as shown in Figures 4a– 4c. The reason is that commodity 802.11ac APs run naive MU-MIMO grouping algorithms that do not consider the inter-user interference, as also corroborated by a recent study [14]. As a result, inter-user interference yields a high SFER, a large number of retransmissions, and reduced PHY rates. But a positive result is that MU-MIMO can reduce the scheduling delay as shown in Figure 4d: with SU-MIMO, each user has to wait on average 75 ms for a transmission, whereas such a delay is reduced by 50% with MU-MIMO. To overcome the performance issues of MU-MIMO when many clients co-exist, the AP should (*i*) make the MU-MIMO grouping algorithm aware of the channel state and/or channel correlation, and (*ii*) dynamically switch to SU-MIMO based on the VR QoE requirements, as described in §3.2.

We then conduct another experiment where 9 clients concurrently upload light data (e.g., the VR meta-data) with and without concurrent bulk download (e.g., the VR content). When concurrent download exists, we observe a significant inflation of the uplink latency, which increases from ~4ms to 13.3/19.8/32.9ms (25th/50th/75th percentiles, respectively). The results demonstrate the need for reducing the uplink delay as discussed in §3.2. We plan to conduct similar performance analysis for 802.11ax when it becomes available on commodity mobile devices in the near future.

5 CONCLUDING REMARKS

Chord enables high-quality multi-user VR on untethered COTS mobile devices over 802.11ac and the next-generation 802.11ax Wi-Fi. Our design consists of both application and wireless layer innovations, whose synergy helps realize the key goals of achieving the scalability, providing good content quality, and embracing clients' heterogeneity. Some of our proposed ideas, in particular those pertaining to the network-level optimizations, can be applied to other bandwidth-intensive applications, such as video conferencing and social AR [17]. We are currently developing the Chord system over commodity servers, enterprise-grade APs, and COTS smartphones.

REFERENCES

[1] 802.11ac In-Depth, Aruba Networks. http://bit.ly/2GAFKTg.
[2] BYOD Popularity. https://goo.gl/1EA1Wf.
[3] Cisco Spark. www.ciscospark.com/VR/.
[4] HTC VIVE for Business. www.vive.com/us/enterprise/.
[5] InsiteVR. https://www.insitevr.com/.
[6] Oculus Go for Business. www.oculusforbusiness.com.
[7] O. Abari, D. Bharadia, A. Duffield, and D. Katabi. Enabling High-Quality Untethered Virtual Reality. In *Usenix NSDI*, 2017.
[8] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu. Motion-prediction-based multicast for 360-degree video transmissions. In *IEEE SECON*, 2017.
[9] K. Boos, D. Chu, and E. Cuervo. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *ACM MobiSys*, 2016.
[10] D.-J. Deng, Y.-P. Lin, X. Yang, J. Zhu, Y.-B. Li, J. Luo, and K.-C. Chen. IEEE 802.11 ax: Highly Efficient WLANs for Intelligent Information Infrastructure. *IEEE Communications Magazine*, 55(12):52–59, 2017.
[11] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han. Rubiks: Practical 360-Degree Streaming for Smartphones. In *ACM MobiSys*, 2018.
[12] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. In *ACM MobiCom*, 2017.
[13] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *ACM MobiCom*, 2018.
[14] S. Sur, I. Pefkianakis, X. Zhang, and K.-H. Kim. Practical MU-MIMO user selection on 802.11 ac commodity networks. In *ACM MobiCom*, 2016.
[15] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu. Supporting Mobile VR in LTE Networks: How Close Are We? *ACM POMACS*, 2(1):8:1–8:31, Apr. 2018.
[16] M. van Der Schaar et al. Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms. *IEEE wireless Communications*, 12(4):50–58, 2005.
[17] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky, and F. Qian. CARS: Collaborative Augmented Reality for Socialization. In *ACM HotMobile*, 2018.
[18] R. Zhong, M. Wang, Z. Chen, L. Liu, Y. Liu, J. Zhang, L. Zhang, and T. Moscibroda. On Building a Programmable Wireless High-Quality Virtual Reality System Using Commodity Hardware. In *ACM APSys*, 2017.