High-performance Intelligent Wi-Fi Multicast For VR/AR Applications

UCLA Wireless Networking Group

Jinghao Zhao, Kaiyuan Chen, Zengwen Yuan, and Songwu Lu
University of California, Los Angeles

Contents

1 Introduction

In recent years, Augmented Reality(AR) / Virtual Reality (VR) have been dedicated to offering users ubiquitous and immersive experiences to users. However, in the context of interactive AR/VR applications, the large bandwidth and low latency requirements of streaming rendered live videos are always challenging tasks for current researchers and software developers. This requirement stems from the unfit of unicast-based design of current Wi-Fi technology: the throughput grows linearly with the number of devices involved , and strongly limits scalability and stability of current AR/VR applications.

As a result, many developers resort to use multicast to solve this problem. By multicast, information can be delivered to a variable group of receivers with only one single transmission, which can be all devices, some of devices or even no device in the group. This greatly increases the flexibility of data transmission. Over a broadcast-based sharing channel, multicast becomes the cornerstone of many wireless protocols. However, trivially applying IP multicast does not solve this problem, because IP multicast still uses unicast in link layer. As a result, real link-layer multicast, which has enabled many new computing and data sharing paradigms in cloud context, should be an ideal transmission medium to AR/VR applications.

Despite of such great demand on link-layer Wi-Fi multicast, current performance on IEEE 802.11 wireless media suffers from packet loss and lack of intelligent mechanisms to support this technique. To renovate current Wi-Fi on this task, we identified the limitations in the existing WiFi protocols and experimented new ways to improve the throughput of wireless multicast over 802.11n protocol. We bypassed many existing limitations of wireless multicast and created working prototypes of Linux and Android platform drivers for Wi-Fi MAC which enable efficient use of multicast transmission and reception at high throughput, and with significantly reduced error rate through adaptive recovery mechanisms. We tested the solution on commercial devices, including a Wi-Fi router, Linux tablet and Android phones. We achieved as high as 110 Mbps UDP throughput (216.7 Mbps PHY rate) in lab, and well over 50 Mbps UDP throughput at the distance of 20 meters. In this document, we present our basic methodologies, preliminary experiment results, and findings that could guide new application scenarios.

We have made the following achievements to support next-generation AR/VR applications:

1. We identified the limitations of the current Wi-Fi unicast-based design for AR/VR applications
2. We designed, implemented, and evaluated the high-performance Wi-Fi multicast for AR/VR
3. We prototyped a preliminary version of enabling learning and reasoning functions to demonstrate the concept of "smart wireless" in the Wi-Fi context
4. We started preliminary integration of the wireless design with the AR/VR application.

2 Limitations of Current Wi-Fi for AR/VR Applications

For the mobile AR/VR applications, we consider the following scenario: an edge server publishes the video stream to a group of devices through Wi-Fi Access Point(AP) The video content can come from another mobile device to support the scenarios such as remote training, or from a graphics server rendering game frames for players. Under this formulation, we identify three main limitations on the current Wi-Fi technology to support AR/VR applications.

2.1 Unfit as Unicast-based design

The current Wi-Fi technology, as stipulated by the IEEE 802.11 standards, is designed to support the TCP/IP suite by providing CSMA/CA based unicast data communications over the wireless broadcast medium. While this design has worked reasonably well for current Internet applications, new issues arise when integrating with the upcoming AR applications.

However, under this context, we find with high-resolution and low-latency requirements of AR/VR applications, wireless unicast does not fit well: not only one has to first identify the remote node to communicate with, but also multiple mobile devices requesting the same data have to be served one by one individually, unable to exploit the broadcast nature of wireless medium.

For example, for the video with 4K resolution and 60FPS, the bitrate needs to reach 60mbps to ensure good quality. The current smartphones with 2x2 MIMO can support up to 300 mbps PHY rate by 802.11n and 866.7 mbps by 802.11ac, which means only 5 and 14 devices can be supported respectively. Due to the noise and other effects from real network conditions, devices cannot sustain at the highest rate all the time. The supported number will be further reduced. The unicast data transmission limits the AR/VR application performance. Moreover, the unicast operation does not work well for the group interactions among AR/VR clients. Furthermore, such multiple unicast communications affect the scaling of the overall solution.

2.2 Inefficiency of IP-layer Multicast

In VR/AR applications, the neither side of the transmission needs to know which specific node to communicate with. That is, AP does not need each individual address to all the devices to the group to perform transmission. As a result, many programmers resort to use IP-layer multicast to reduce the problem of addressing, and pertain a misconception that IP multicast can solve the poor scalability of traditional unicast design.

Intuitively, IP-layer multicast is sufficient for high throughput and low latency transmission. This conception is true for wired network. For the current video streaming applications like IPTV, it uses IP-layer multicast to deliver the high-resolution video on a large scale. The wired network has wider bandwidth and lower loss rate, so applications can reach decent performance. However, this conception is wrong in the context of Wi-Fi. When the IP-layer multicast packets come to link-layer for transmission, they will be converted into unicast frames, which is essentially the same as the aforementioned unicast-based design and thus the advantage of IP-layer multicast is eliminated. As a result, the inefficiency of current multicast protocol stack extremely limits the mobile AR/VR applications.

2.3 Lack of Learning and Reasoning Capabilities

In addition, the current Wi-Fi lacks intelligence on providing "what" or "why" answers for both normal operations and failure handling. This becomes a major problem for system managers and application developers on AR/VR applications. The current wireless design adopts a *blackbox* design approach. Consequently, the higher-layer protocols and applications have to guess what is going inside the wireless network based on their own observations. This definitely limits the operation efficiency and application performance.

As a result, we identify three issues regarding the intelligence of the current Wi-Fi technology: poor learnability, weak information exchange, and poor reasoning capability. These problems greatly hinder the design of wireless system, making AR/VR applications hard to detect and

respond to mobility, failures and wireless dynamics.

The current Wi-Fi largely operates as a blackbox for higher-layer designs, in both the current AP management and smartphones. Low-level MAC & PHY information cannot be readily acquired and analyzed by phone users, and users thus do not know how and why failures arise or mobility is triggered. In addition, it is hard for both the AP managers and phone users to know where things go wrong. For example, low throughput can caused by either user mobility or by AP misconfigurations. However, in the current Wi-Fi operations, these conditions cannot be acquired and informed to each other.

Since the current Wi-Fi does not enable exchanges on important information, AP and phone users are virtually blind to each other. AP can acquire some basic user information such as device RSSI, but it lacks valuable information such as mobility and signal strength relative to other APs. Such information is critical to applications such as interactive AR and real-time streaming applications, which require high throughput and stable channel conditions. Even if the current AP design may collect some device-side information such as device RSSI, it cannot perform some basic reasoning tasks given the collected information, and make intelligent decisions (for example, to perform multicast roaming between different APs).

3 Case for High-performance Intelligent Wi-Fi Multicast For AR/VR Applications

The aforementioned limitations on using multiple transmission to transmit one data packet strongly limits the scalability of AR/VR applications. When bandwidth and latency are bottlenecks to the system, efficient and effective usage of shared channel is the key to solve this problem. We thus make a case for Wi-Fi multicast as the basic wireless access solution to AR/VR applications. This is motivated by the following factors.

Real link-layer Wi-Fi multicast makes use of broadcast nature of wireless medium to enable multiple nodes share the videos together, thus improving scalability. The next-generation AR/VR become possible by leveraging the advantages of real multicast.

High-performance Wi-Fi multicast with sufficient rate can serve the high-resolution videos (for example, 4K or 8K video streams), which is highly needed for the AR/VR applications. While the unicast-based solution may not be able to offer all users with the high-fidelity video streaming, Wi-Fi multicast can support interactive AR/VR applications with high quality and trigger new scenarios.

In addition, with the capability of performing learning and reasoning functions over Wi-Fi, we can leverage available cross-layer information at the client device, and the wireless AP can learn the current health status at each client and take adaptive actions accordingly. We have applied this idea to handle device mobility upon roaming and cope with network failures. When combined with the multicast design, the AP know the runtime status of each client and take corresponding reactions. As a result, Wi-Fi is enhanced with higher degree of intelligence.

4 Issues with Current Wi-Fi multicast

Limitation of Wi-Fi stems from two major sources: IEEE 802.11 Wi-Fi standards and current Wi-Fi multicast implementation on commodity devices.

4.1 Limitations of Wi-Fi multicast in the current Wi-Fi standards

The legacy Wi-Fi does support multicast delivery in its IEEE 802.11 standards. However, we have identified the following five limitations with such solutions:

1. The multicast rate is locked at the lowest speed Currently wireless multicast is mainly used to deliver management traffic over Wi-Fi, it only runs at the lowest base rate (called basic rate). By default, this is typically the lowest speed supported by the AP, for example, 1 Mbps at PHY. While this low rate ensures that the broadcast/multicast frames from the AP can be received by all clients, it is usually too low to meet applications' needs.

2. The 802.11 power-save mode further limits the multicast throughput When a wireless client associated with an access point (AP) enables its 802.11 power-save mode, the AP buffers all multicast frames and sends them after the next DTIM (Delivery Traffic Indication Message) beacon, which may be every one, two, or three beacons (referred to as "DTIM interval").

The use of DTIM intervals was included in the 802.11 standards to allow the sleeping clients that implement the power-save function to learn when to wake up and receive the multicast traffic (i.e., after every DTIM beacon) and to enable flexible configurations (i.e., DTIM interval) to balance between battery life and throughput performance. This performs well for energy efficiency, but limits the maximum achievable throughput for multicast traffic.

3. Frame aggregation is not enabled for multicast frames Frame aggregation is a feature of the recent IEEE 802.11n/ac standards, that increases throughput by aggregating multiple individual frames into a single large super-frame. It amortizes the frame header overhead, airtime for inter-frame spacing and acknowledgments of individual frames.

Through detailed code analysis and experiments, we have concluded that multicast frames are not sent with frame aggregation in the default vendor implementation. In the regular operation, an A-MPDU frame is followed by a selective block ACK. However, since multicast does not support ACK mechanism, the A-MPDU aggregation is not supported for multicast.

4. Higher channel width is not enabled for multicast frames In wireless transmission, utilizing more channel width would help to reach higher throughput. In the default setting, multicast over 802.11n runs on the 20 MHz channel at 2.4 GHz only. It does not support operations over the 40 MHz channel at either 2.4 GHz or 5 GHz.

5. No retransmission mechanism for multicast The existing Wi-Fi multicast is mainly used to transmit the control frames rather than data frames. Therefore, it only provides best-effort transmissions. Thus, it is deemed inefficient for data delivery and cannot be used for application data transfer. The research community has known such issues for years [3].

4.2 Limitations of available Wi-Fi multicast implementations

Due to the limitations of legacy Wi-Fi multicast standards, vendors use alternative solutions. The first option is to use multiple unicast, which transfers the multicast frames using multiple unicast deliveries to all involved clients. This solution will meet the bottleneck when there are many devices in the multicast group. With unicast, the AP can only send the data to one device at a time. With many devices, the average throughput for each device is thus limited. It thus limits the benefits of true multicast.

The second option is to enable true multicast at the link layer. However, such hacking solutions [1] [4] only work with the early generation of Wi-Fi devices, such as 802.11a/g. They can support multicast rate up to 54Mbps at the physical layer, but cannot work with the latest

802.11n/ac devices, thus unable to support higher multicast rates higher, say 100s Mbps.

The next observation is that, current multicast solutions do not work with commodity smartphones. They work on mobile devices running Raspberry Pi or Linux. However, mobile users usually prefer to running their applications on their smartphone devices. This poses as another limitation of the current vendor solutions.

5 Resilient Link-Layer Multicast in Wi-Fi

Given the devices for most of AR/VR applications, our objective is to implement a standard-compliant link-layer multicast solution that works with current commodity APs and commodity Android smartphones.

5.1 Goals

We seek to achieve three concrete goals:

1. The multicast solution must operate at the link layer and comply with the current Wi-Fi standards.
2. The solution must support high rate multicast delivery to meet the demand for AR/VR applications.
3. The implementation must work with current Wi-Fi chipsets and can run on commodity Android phone devices.

5.2 Our Proposed Solution Overview

To address the above issues, we came up with the following solution techniques to tackle each issue item-by-item. We jointly design the AP and mobile phones to enable high-rate, intelligent Wi-Fi multicast. On enabling high-rate multicast, we enable high rate multicast up to 450 Mbps, eliminate idle interval caused by DTIM buffer, enable A-MPDU frame aggreagation, expand multicast operation mode for 40MHz and design a L2.5 retransmission mechanism. In addition, we open up the black-box design of Wi-Fi by devising a feedback mechanism between AP and devices with learning and reasoning capabilities.

5.3 High-Rate Multicast

1. Enabling high-rate multicast up to 450Mbps The current Wi-Fi multicast runs at the lowest supported data rate. The available hacking implementations only increase the multicast rate within the 802.11a/g standard, which can only be up to 54Mbps. The more recent 802.11n/ac standards use MIMO and higher MCS (modulation and coding scheme) rates. However, both MIMO and higher MCS rates cannot be supported by current hacks. To this end, we further modify the device driver at the AP side to support higher-rate MCS choices for multicast. By enabling such high rates, we can reach up to 450Mbps for multicast frames over 802.11n.

2. Eliminating the idle interval caused by DTIM buffer We have observed that in the device driver implementation, the DTIM buffer incurs the under-utilization of the wireless channel when sending out multicast frames. By the default setting, it sends 128 multicast frames every 2

beacon intervals (200ms). However, the router actually spent less than 200ms to send out those 128 frames, creating an 200ms idle period between every two beacons.

We seek to keep the router transmitting as many multicast frames as possible. Therefore, we transmit continuously the multicast frames to maximize the throughput by ignoring the DTIM buffer.

3. Enabling frame aggregation (A-MPDU) for multicast By enabling frame aggregation for the multicast frames, we let the multicast frames be aggregated into super-frames for transmissions. Through frame aggregation, we can amortize the overhead caused by inter-frame space and frame headers, thus reaching much higher throughput.

4. Expanding multicast operation mode for 40 MHz channel width In the current 802.11n Wi-Fi, if we increase the channel width from 20MHz to 40MHz, the maximum physical-layer (PHY) rate should double with the same MCS rates.

We thus seek to enable the 40MHz channel operation for the multicast frames. While it is easy to set the 40 MHz channel width running in the 5GHz frequency spectrum, there is a challenge when running it at 2.4 GHz frequency it is a practice commonly advised against due to its interference with other channels. In fact, many Wi-Fi devices do not even support such an operating mode, e.g., in both the latest MacOS and Android systems. Even on the tablet running Ubuntu 14.04, the throughput does not improve obviously.

We have also tried with different combinations of available channels (channels 111) at 2.4 GHz in order to bind a 40MHz channel. The results show that, different channel bonding exhibits different throughput result. To address this issue, we conclude that running high rate multicast at 5 GHz is the better choice.

5. Devising L2.5 resilient mechanism for robustness The curren Wi-Fi multicast does not use any retransmission mechanism. The key components for the retransmission mechanism include Sequence Number, Acknowledgement, and Retransmission. To increase the robustness of Wi-Fi multicast, we have designed resilient multicast mechanisms based on the current 802.11n standard. To this end, we have added a custom L2.5 header, which sits between Layer 2 and Layer 3. We consequently manage the multicast frame retransmissions using this new header.

(A) L2.5 Header Structure The L2.5 Header is shown in Figure 1. It has four parts: Sequence number, GroupID, Frame Type, and Bitmap.

1. For the DATA frames, the Sequence Number at L2.5 Header is similar to the sequence number at 802.11 frame header, which is a 12-bit number used to number frames sent between AP and clients. For the type BlockAck, the Sequence Number represents the start sequence number in the corresponding window.

2. The GroupID is used for distinguishing multiple multicast groups. For now, it contains two bits, which can represent up to 4 different multicast groups. The value is set to 0 by default.

3. Frame Type is represented by two bits. 00 indicates DATA frame, 01 indicates BlockAck frame, 10 indicates retransmission DATA frame and 11 are reversed in case we want to extend the type in the future.

4. The Bitmap contains 64 bits to indicate the state of continuous 64 frames from the Sequence Number. This field is only included in Block Ack to inform AP which frames are received by the client. AP can aggregate Bitmaps from different stations to retransmit lost frames.

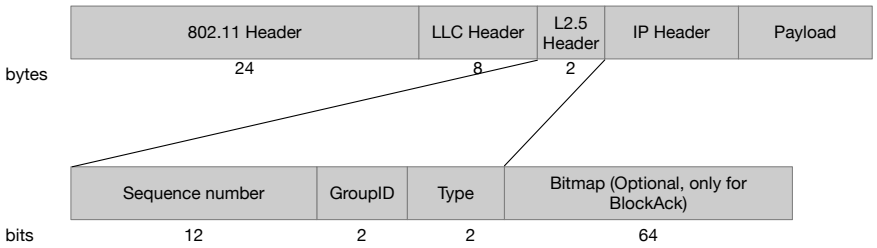


Figure 1: L2.5 Header Structure

(B) Procedure of Resilient Wi-Fi Mulitcast The basic procedure is illustrated in Figure 2. Upon receiving aggregated multicast frames sent from the AP, each client tracks the Sequence Number in the L2.5 Header, and records whether each individual frame carried by this aggregated frame has been successfully received or not. After receiving multiple aggregated multicast frames (64 frames in current design), the clients unicast the BlockAck message back to the AP, which include the bitmap to indicate the status of each individual frame received during this interval. Since the BlockAck is a unicast frame, it will receive the acknowledgment from the AP. The current Wi-Fi standard provides the retransmission mechanism for the unicast frames.

Upon receiving the BlockAcks from clients, AP schedules to retransmit the lost frames for each client. By setting the maximum number of retransmissions, we avoid the deadlock and keep the frame queue moving forward. We also use this mechanism for future rate adaptation and traffic classification. For example, for video streaming, we can set a larger retransmission limit for the important frames so that these frames will have higher reliability.

Upon receiving a retransmitted DATA frame, a client would check whether it has received this frame before. For the lost frame, the client drivers will deliver the frame to the upper layer; otherwise, it will drop the frame to reduce the overhead for further processing.

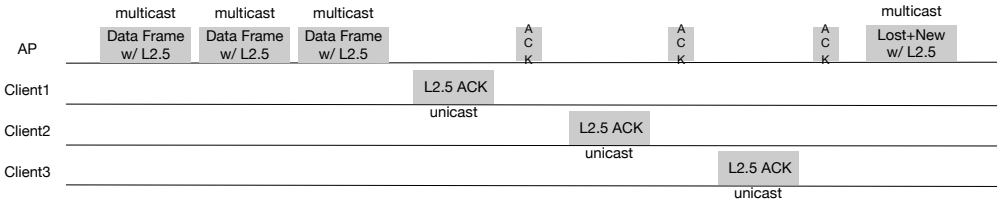


Figure 2: Procedure of Resilient Wi-Fi Mulitcast

5.4 Intelligent Multicast

We now describe our prototype efforts on the smart Wi-Fi that enables the Wi-Fi with learning and reasoning capabilities. Given that the current Wi-Fi technology operates as a black-box system to the TCP/IP protocol stack, we make Wi-Fi more intelligent so that it learns from cross layer Wi-Fi information and side information at the client, and reasons on network failures and mobility. The client then communicates its learning and reasoning results to the AP, which consequently enhances the multicast performance of critical services such as roaming between different APs.

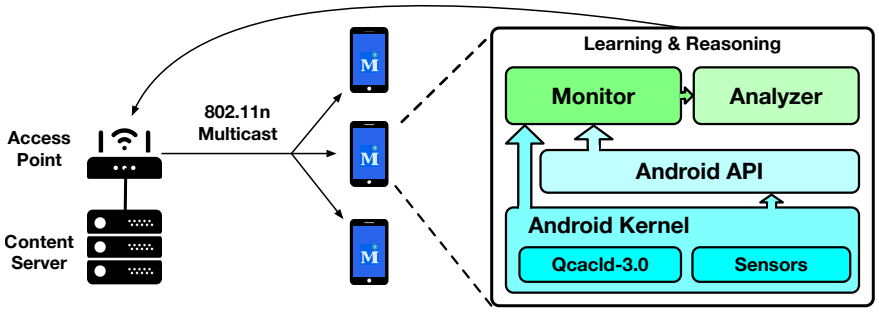


Figure 3: A basic design of Intelligent Wi-Fi

Our Design

In our design, we open up the network information access to the blackbox Wi-Fi, and conduct data analytics to enhance critical services such as multicast roaming between different APs. Consequently, we can improve the reliability and performance of AR applications. A basic design of Intelligent Wi-Fi is shown in Figure 3. We adopted the same monitor-analyzer framework as MobileInsight [2], which is a widely used community data analytics (300+ universities and companies to date) designed by us.

We first design our learning module by exposing raw logs from different sources to the device user-space. For our essential Wi-Fi logs, it requires modifying kernel Wi-Fi drivers. The learning module parses the raw logs and extracts the carried information. It also learns other side information from system APIs, such as mobility, and remaining battery level, to facilitate learning.

Given the learned information from heterogeneous sources, our reasoning module is designed to reveal the operational dynamics behind the Wi-Fi multicast. Based on the learned information, we perform reasoning by exploiting previously learned results and domain knowledge. The reasoning module infers various conditions, such as weak channel condition, and mobility. The inferred results are sent as feedback messages to the AP to further facilitate the AP make intelligent decision.

Upon receiving the conditions from the phone users, the AP can make decisions based on such knowledge. It manages to send responses to phone users to notify its decision and help phone users to achieve better performance on ICN applications.

6 Implementation

6.1 Device-Driver Prototype at both AP and Android Phones

We have implemented our high-rate and resilient link-layer Wi-Fi multicast on multiple device drivers of wireless AP and clients. They include Linux mac80211 kernel module, ath9k driver for Qualcomm Atheros AP, mwifiex driver for Microsoft Surface Tablet, and qcacld-3.0 driver for Google Pixel2 smartphones that run Android. Our designs can be widely used for multiple mainstream drivers, especially for Android smartphones. We plan to release our source code once they are further tested and stablized.

Based on our prototype, we have further constructed a small experimental testbed for validation and evaluation. Our testbed has two APs and three client devices, all of which run commodity 802.11n MIMO. The AP is a TP-Link N750 router (Wi-Fi chip: Qualcomm Atheros AR9580). STA1 is a Microsoft Surface tablet, running Ubuntu 14.04.5 LTS. Its WLAN chipset is Marvell's

Avastar 88W8897 SoC. STA2 and STA3 are Google Pixel 2 phones, running Android 9.0. Its WLAN chipset is Qualcomm Snapdragon 821 SoC.

6.2 Application Layer Design

We have implemented the above design on Android Phones as the clients and TP-Link N750 router as the wireless AP. At the phone side, we modify Qcacld 3.0 driver to print out more information and replace the original Pixel 2 driver by Magisk. We implement our Android learning module to collect driver logs and Android system APIs. The application also collects other side information such as battery level and the status of mobility (static, walking, etc.), and analyzes the collected information by performing reasoning tasks. Once reasoning at the phone is completed, it sends out feedback messages to the AP. In our current prototype, we consider the influence of mobility of smartphone users, and the channel strength to the multicast frame loss rate. A sample screenshot can be found in Figure 4.

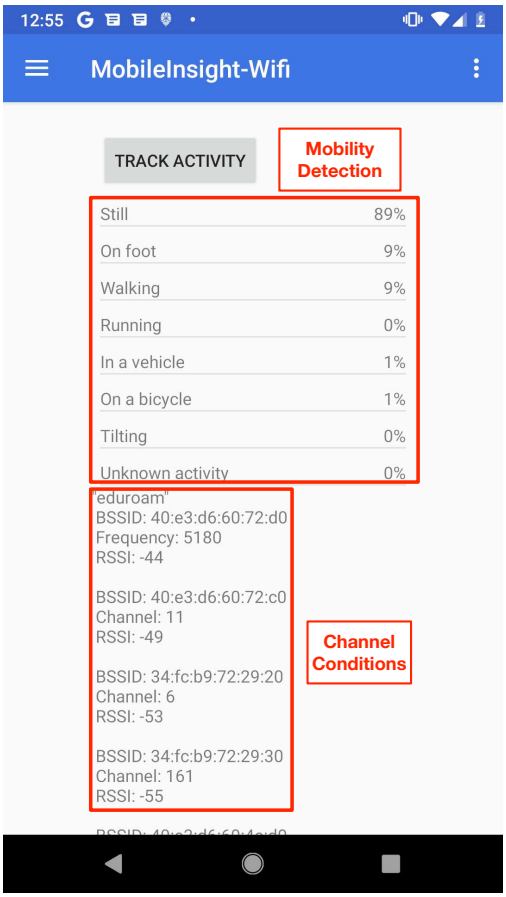


Figure 4: A Screenshot of Our Prototype Phone Application

Since user mobility can lead to varying signal strengths and greatly influence the multicast receiving rate, we use the Google Android Mobile Service API to perform activity recognition transition predictions, which can classify the current user as still, on foot, walking, running, in a vehicle, on a bicycle and tilting. Once we detect the user as not being still or on foot, we send a feedback message to the AP to inform the current user's unstable state. As a result, the AP will not consider this user in further decision making when selecting the multicast sending rate. In

order to notify the user of its decision, the AP sends a response message to the client and informs the user. A screenshot of our notification can be found in Figure 5.

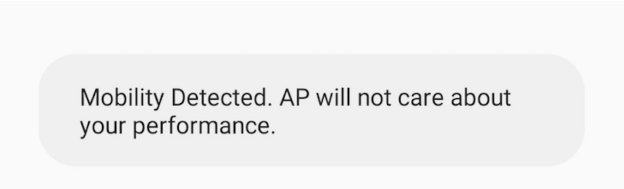


Figure 5: A sample notification of AP's decision

A similar procedure can be applied to the signal strength case when there is no roaming during multicast. In order to learn the channel information, we use the Android system API, WifiManager, to learn the signal strength on different channels. When there are multiple APs over different channels with different RSSI, the intelligent Wi-Fi can learn its relative signal strength to all other APs with the same SSID, which it performs multicast roaming to them. Thus, it sends a feedback signal to the currently connected AP, thus informing its current condition. Once the AP receives the feedback message, it can make the decision on suggesting the current user roam to other APs, depending on its knowledge about other APs, such as other APs rate configuration and traffic load.

One of the important metrics for users multicast performance is the frame loss rate. We acquire it using our resilient mechanism. Resilient mechanisms can provide more low-level info from the qcacld-3.0 driver. In our resilient mechanism design, the driver is required to record the status for all the multicast frames and report the L2.5 BlockAck to the AP. At the same time, we can analyze the real-time status to infer the current network condition. By learning from the low-level frame loss rate, the application can reason the current conditions, e.g., there are some burst drops due to network instability, or the network is experiencing a bad situation for a long period. The application can report the related information to the AP, so that the AP can make corresponding decisions and the multicast performance will not be affected by any individual device that underperforms.

For the reasoning module at the AP side, we modify Ath9K driver to filter out feedback messages from all devices. We leverage the built-in Lua support of the AP to implement a script to parse the feedback signals. We further perform AP side-reasoning and decision making by Lua script, and send a response signal back to each Android phone. The response message includes suggestions on mobility or roaming to other APs with better channel conditions. Since the AP can receive multiple reports from different layers of user's equipment, the AP thus makes decision on based on the available cross-layer information.

6.3 Integration with the AR Application

We now describe our efforts on integrating the wireless prototype with the AR application. Our integration effort involves integrating the AR/VR modules with our two wireless function prototypes: the high-rate, link-layer multicast and the smart Wi-Fi with both learning and reasoning capabilities.

Integrating AR/VR with link-layer multicast To integrate with our resilient link-layer multicast, the AR/VR application uses the multicast to enhance the AR/VR performance. For the phones running the AR/VR application, they will multicast their contents, so that multiple servers can process and store the contents at the same time. Meanwhile, the phones as clients will request

the data from the servers. and the servers will multicast requested data to all those clients in the group. This demonstrates that multicast is more efficient for AR/VR applications over wireless.

The available vendor-specific Wi-Fi multicast, which is constrained to transmit at the low base rate, cannot meet the bandwidth requirements for multicasting the video data to all clients. However, by leveraging our resilient Wi-Fi multicast, which enables multicast the video data at high rates and with retransmissions, the AR application can stream the video frames to the interested clients smoothly. This integration has been completed with the AR/VR application.

We have implemented our design on the driver of AP (Ath9k for TPlink N750) and clients (Qcacld-3.0 for Google Pixel 3 XL with latest Android 9.0) in the AR/VR system. The results show that Resilient Wi-Fi Multicast can work in concert with the rest of the AR/VR system.

In our prototype, the AR/VR application runs on the latest Android 9.0, which imposes more restrictions such as MulticastLock for the application to receive the multicast traffic. We bypass such restrictions by integrating the related functions into the AR/VR applications. Through our direct integration with the application, the AR application is multicast friendly. The application can continuously receive the multicast data, while it displays AR video streams smoothly. As a result, our resilient Wi-Fi multicast meets the design expectation for the support of next-generation AR/VR applications. The AR application integration with our high-rate, link-layer multicast has been completed.

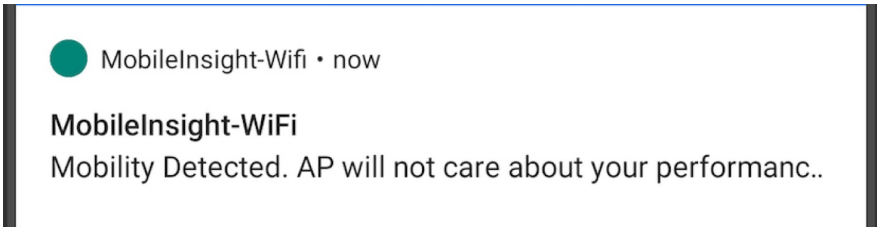


Figure 6: A sample notification for the AR application

Integrating smart Wi-Fi with AR/VR application We further integrate our intelligent Wi-Fi designs, together with the link-layer multicast, into the current AR/VR application prototype. Since the AR/VR application is based on the high channel bandwidth and mobility requirements, we need our current Wi-Fi multicast system to adapt to more scenarios and to identify the current failures for both the AP and the clients. As a result, we install our standalone Android application module into the current devices. Once the multicast frame delivery rate is low, we start to analyze the root cause at the AP side, in order for AP to make more intelligent decisions. For example, in Figure 6, we show the notification prompted in Android Phone when we run the AR/VR application, which informs the phone users of the current state of the AP and how the user should react to improve the performance of the Wi-Fi multicast.

Through our cross-layer integrations among the device driver, OS, and application, our resilient link-layer multicast and smart Wi-Fi with enhanced intelligence have been integrated with the AR/VR system.

7 Evaluation

In this section, we use experiments to demonstrate the performance of our design under different settings.

7.1 Testbed and Experimental Setup

In this subsection, we describe the testbed that we use and the settings of our experiments.

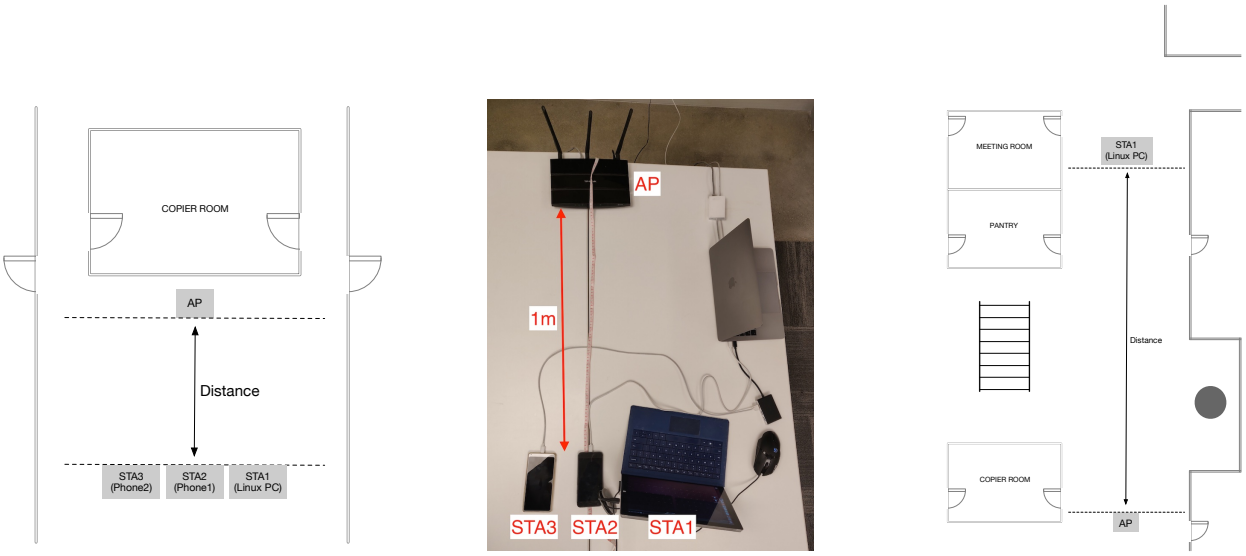
Testbed

For the AR/VR applications, multiple users can share the high-resolution videos at the same time. Our results show that, resilient multicast can support 8k video streaming for AR/VR applications. A desktop computer, which serves as the content server, connects to the AP and multicasts 8k video to clients in the multicast group. Our prototype captures the real-time screen from a camera of the smartphone and streams the video to the RTSP server. After transcoding the video, the server further streams the real-time video to multiple users through our resilient Wi-Fi multicast. We have confirmed that, our prototype on resilient multicast can well support real-time streaming for industrial-grade AR/VR applications.

We conducted extensive measurements to gauge the impact of each parameter setting on the throughput and loss performance of our high-rate Wi-Fi multicast. We now describe the common experiment setting, and then introduce the specific experiments.

The device and experimental layout are as follows. We test our link-layer multicast over 802.11n between one AP and three clients. The AP is a TP-Link N750 router (Wi-Fi chip: Qualcomm Atheros AR9580). STA1 is a Microsoft Surface tablet computer, running Ubuntu 14.04.5 LTS. Its WLAN chipset is Marvell's Avastar 88W8897 SoC. STA2 is a Google Pixel smartphone, running Android 8.0. Its WLAN chipset is Qualcomm Snapdragon 821 SoC. STA3 is a XiaoMi MIX2 smartphone, also running Android 8.0. Its WLAN chipset is Qualcomm WCN3990.

The AP is placed the same height as the three clients, lined up as shown in Figure 7a. In the experiments, the AP and STAs are placed within short distance (within 5 meters), as shown in the setup layout of Figure 7a. For longer distance (say, larger than 10 meters) experiments, we use the setup layout of Figure 7b.



(a) Short-distance experimental setup (1m5m)

(b) Larger-distance experimental setup (10m20m)

Figure 7: Experimental Setup

In the tests, we use iPerf2 to multicast UDP packets. The UDP packet length is 1470 bytes (to reach the MTU of an Ethernet frame). The multicast packets are sent to the multicast group with an address of 224.5.0.8 (IP multicast address) and 01-00-5E-05-00-08 (Ethernet multicast address).

To disable the power-save mode at clients (specifically for Android phones), we use low-throughput background ping traffic to keep them awake (ping -i 0.05 [router_IP]). We test three rounds for each experiment setup. We send out UDP traffic for 60 seconds in each round.

Metrics

We compute and compare the following metrics in each of the three experimental runs:

1. **UDP throughput**: Based on the AP settings and parameter, we can look up the 802.11 standard for the maximum PHY throughput. However, it does not represent the true data throughput seen at higher layers, given the layered protocol header overhead, inter-frame air time overhead, and ACK overhead. Therefore, we use iPerf2 to compute the actual data throughput achieved by the UDP traffic. It will be calculated as the average data throughput over all 3 runs.

2. **Byte loss rate** Because of channel interference and lack of the ACK mechanism, received frames at the clients can be error prone. To quantify the loss rate, we compare the number of bytes from the AP TX queue and the number of bytes from each client's RX queue over the same period of time. Such statistics are provided by Linux kernel and available at /proc/net/dev. We define Byte Loss Rate as:

$$Byte\ Loss\ Rate = Average\{\frac{\#\text{ of bytes sent from AP}'\text{s TX queue}}{\#\text{ of bytes sent from STA}'\text{s RX queue}}\}$$

3. **Frame loss rate** Similarly, the loss may be quantified by the number of lost frames. The statistics are also available at /proc/net/dev. We define the Frame Loss Rate as:

$$\text{Frame Loss Rate} = Average\{\frac{\#\text{ of frames sent from AP}'\text{s TX queue}}{\#\text{ of frames sent from STA}'\text{s RX queue}}\}$$

4. **Data loss rate** Finally, we use the built-in data loss rate in iPerf to further illustrate the real data transmission loss. Note that, this metric is not meaningful once the sending rate exceeds the maximum data rate that multicast can support.

In our experiments, we use the following parameters as the default setting unless explicitly specified:

- AP and STAs are running at 802.11n mode, 5 GHz
- Bandwidth: 20 MHz (default) or 40 MHz
- Channel: channel US-153 (default) or variable
- AP TX power: 21 dBm/125 mW (default) or variable
- Multicast MCS index: MCS 15 (default, corresponds to 5/6 coding rate, 64-QAM, 130 Mbps PHY rate) or variable
- AP and STA distance: 1 meter (default) or variable
- Sender throughput: 100 Mbps or variable

- A-MPDU length Limit: 30000 bytes (default) or variable
- UDP packet length: 1470 bytes (default) or variable
- MIMO: 2x2 (default) or 3x3

For each test, we only vary a parameter at a time and fix other parameters. We then use iPerf2 to conduct three test runs (each takes 60 seconds). Finally, we compute all metrics and show our results. We test the following items:

1. Throughput and loss by varying sender throughput
   We send varying sender throughput: 20 Mbps, 30 Mbps, 40 Mbps, ..., 120 Mbps.
2. Throughput and loss by doubling channel bandwidth We compare throughput and loss when using 20 MHz and 40 MHz channels.
3. Throughput and loss by varying distance We vary the distance between AP and STAs: 0.5m, 1m, 1.5m, 2m, 2.5m, 3m.
4. Throughput and loss by varying AP TX power We vary TX power levels: 0 dBm, 5 dBm, 7 dBm, 10 dBm, 15 dBm, 20 dBm.
5. Loss by varying MCS index and receiving distance We set varying MCS indexes 11, 12, 13, 14, 15 at distances 10m, 15m, 20m.
6. Throughput and loss by varying A-MPDU length limit We set varying A-MPDU lengths: 2500, 5000, 7500, 10000, 20000, 30000, 40000, 50000.
7. Throughput and loss by varying UDP length We send varying UDP length: 500, 1000, 1470 (which add header equals to the max frame length 1500 that will not be fragmented), 2000.
8. Throughput by changing MIMO settings We test both 2x2 MIMO and 3x3 MIMO.
9. Resilient Mechanism We test the performance with and without the resilient mechanism.

7.2 Solution Validation

We first validate our prototype, which made changes and patches to the mac80211 driver and ath9k drivers. Our assessment is on a TP-Link N750 router. We use iPerf2 to multicast UDP data packets to IPv4 multicast address 224.5.0.8. We then wirelessly connected three clients to the router. One is a tablet computer running Linux, and the other two are Android phones running Android 9.0. All can correctly receive multicasted data simultaneously.

To further validate that our Wi-Fi multicast is running with the desired parameters, we use another MacBook Pro laptop as a sniffer, and run Wireshark monitor mode to sniff over-the-air data transmissions. We successfully intercept all the data being multicast, as shown in Figure 8.

We examined the captured frames in Figure 8. Our analysis shows that, it is indeed an ongoing UDP multicast session, without data duplication (so it does not use multiple Layer-2 unicast sessions for the UDP multicast). The wireless transmission is using 802.11n, at 5 GHz band (Channel 40) with an 40 MHz channel width. The maximum MCS index we can set is 23. With short GI being used, the PHY rate is 450 Mbps. The destination MAC address (01-00-5E-05-00-07) is indeed a multicast address according to the Ethernet IPv4 Multicast address assignment (RFC 1112).

In summary, we conclude that, our solution is implemented correctly. We describe the test setup and results next.

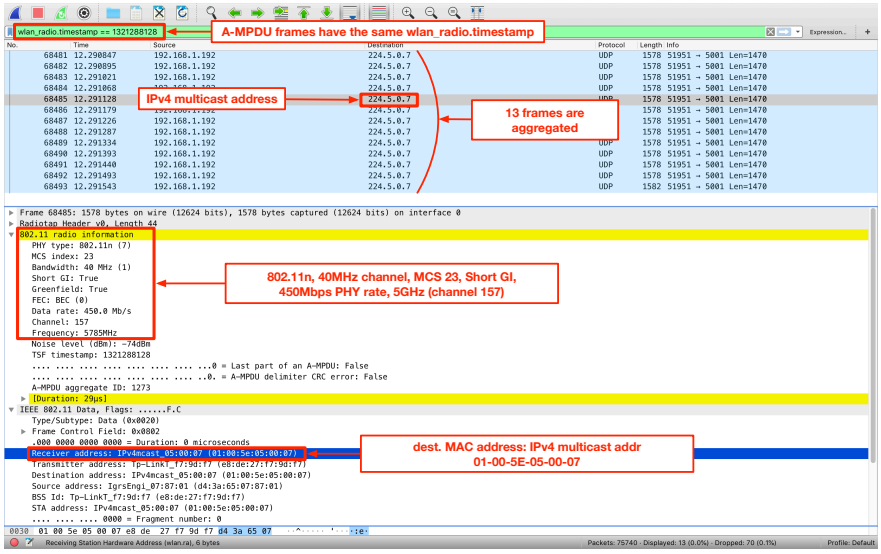


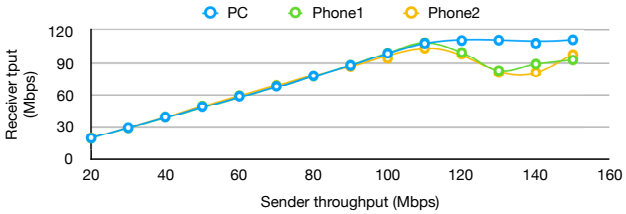
Figure 8: Captured over-the-air multicast transmissions in the Wireshark monitor mode

7.3 Experiment Results

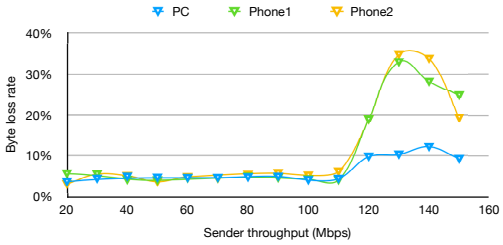
We now show the experimental results according to the above plan.

I. Varying sending throughput

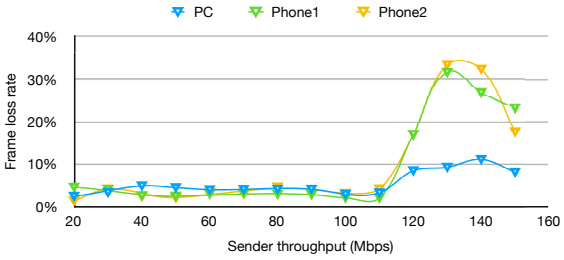
We test the maximum UDP throughput our solution can achieve under the default settings, and also how receiver-side throughput and loss changes as the sender-side rate increases. We set multicast PHY rate to 144.4 Mbps given MCS index 15 and Short GI, and set other parameters as the default values.



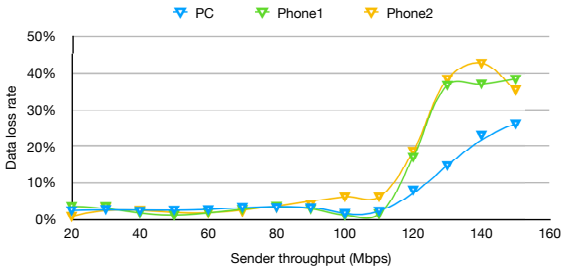
(a) UDP throughput of varying sending throughput



(b) Byte loss rate of varying sending throughput



(c) Frame loss rate of varying sending throughput



(d) Data loss rate of varying sending throughput

Figure 9: Varying sending throughput

(1) UDP throughput: Figure 9a shows the maximum achievable UDP throughput is around 110 Mbps using multicast at 1 meter distance. All three clients' throughput grows until the sending rate reaches 110 Mbps. After that, the PC achieved stable data throughput at 103 Mbps, while throughput of two Android phones dipped.

(2) Byte loss rate: Figure 9b shows the byte loss rate is steadily around 5% using multicast at 1 meter distance between AP and STAs. Note that if the sender throughput exceeds the maximum value (at around 110 Mbps), due to the congestion and receiving capability of phones, the byte loss rate would increase.

(3) Frame loss rate: Figure 9c shows that the frame loss rate is also the same, if not lower, as the byte loss rate at around 4%. The frame loss rate also exhibits similar behavior when sender throughput exceeds the maximum value.

(4) Data loss rate: Here we show the UDP data loss rate measured by iPerf2. Before the throughput reaches maximum, the data rate is around 3%. Note that the data points after 110 Mbps is not meaningful, as the data sent out by the sender is limited to 103 Mbps but the receiving side at iPerf2 does not take it into consideration.

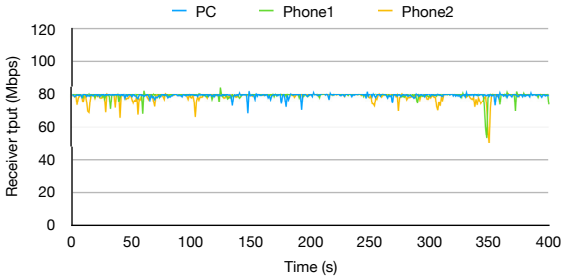
(5) Demystifying throughput dip at clients: Since we observed different behaviors on PC and Android phones when the sending throughput reaches maximum, we further conducted tests. We tested three rates of 80 Mbps, 100 Mbps, and 120 Mbps, to show the differences between the receiver throughput when the sender throughput is (a) lower than the maximum throughput, (b) nearly at the maximum throughput, and (c) higher than the maximum throughput.

We record the throughput at clients every second, and plotted the throughput-time curve. When the sender throughput is below or at the maximum throughput, all clients can sustain stable throughput; however, when the sender throughput exceeds the maximum, due to the congestion and receiving capability of phones, there will be periods of throughput drops, thus averaging down the overall throughput shown. We tested 400 seconds for all three cases as shown below.

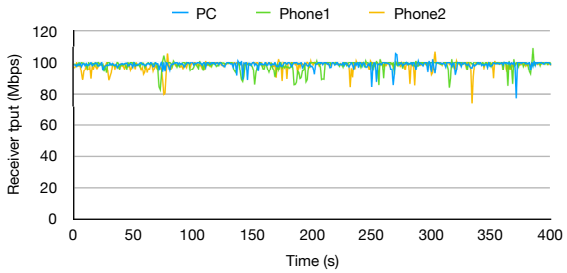
(a) Sender throughput = 80 Mbps (lower than the maximum throughput)

(b) Sender throughput = 100 Mbps (nearly at the maximum throughput)

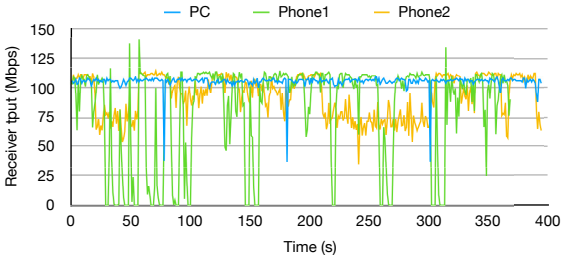
(c)Sender throughput = 120 Mbps (higher than the maximum throughput)



(a) Sender throughput = 80 Mbps



(b) Sender throughput = 100 Mbps



(c) Sender throughput = 120 Mbps

Figure 10: Demystifying throughput dip at clients

II. Varying channel width

In this experiment, we gauge the effect of doubling the channel width. We increase the channel width from 20 MHz to 40 MHz and fix other parameters. We find that, the sustainable max throughput doubles when the bandwidth doubles on both 2.4 GHz and 5 GHz frequencies before the sender throughput reaches the maximum capacities of clients.

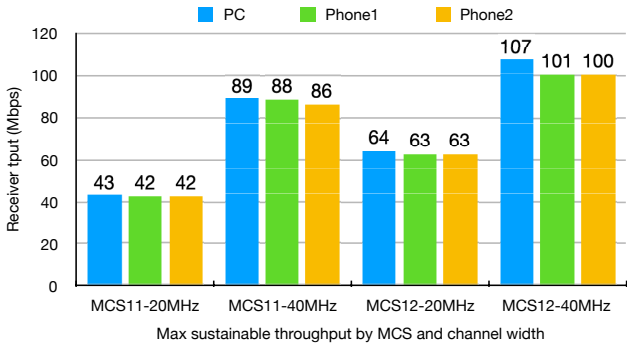


Figure 11: Receiver throughput of varying channel width

We compared throughput performance by doubling channel width using two MCS indexes, as shown above. When we used MCS 11 (max. PHY rate = 52 Mbps at 20 MHz and 108 Mbps at 40 MHz), the maximum UDP throughput did double from 43 Mbps (at 20 MHz) to 88 Mbps (at 40 MHz), demonstrating the effectiveness of increasing channel width.

However, when we further increased the MCS to 12 (max. PHY rate = 78 Mbps at 20 MHz

and 162 Mbps at 40 MHz), the maximum UDP throughput did not double. In fact, it merely increased from 63 Mbps (at 20 MHz) to 101 Mbps (at 40 MHz). Our analysis show that it is possibly caused by the capacity of the router and the stations. In the results shown in F.I, the maximum UDP throughput we can achieve is around 110 Mbps. Exceeding such rate will not yield higher throughput. Even though doubling the channel width using MCS 12 would have theoretically doubled the throughput to 126 Mbps, it has already exceeded the maximum throughput we can sustain. Therefore, we did not see throughput doubling using MCS 12 or higher in our tests.

III. Varying receiving distance

In this experiment, we test how increasing the receiving distance affects throughput and loss. We varied the distance between the AP and stations and fixed other parameters as default.

(1) UDP throughput: Figure 12a shows that, the throughput does not vary much when the receiving distance varies. At 100 Mbps sending rate, the achieved throughputs on all stations are around 98 Mbps as the distance increases from 0.5m to 3m. It could be caused by the maximum TX power we were using. The throughput test for longer receiving distance is described later.

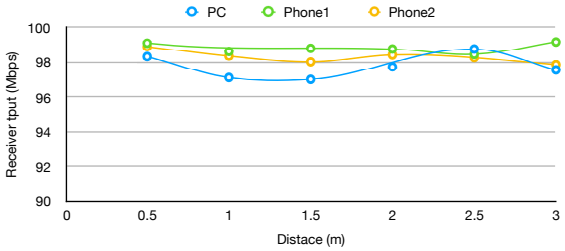
(2) Byte loss rate: byte loss rate remains below 4% as the distance increases from 0.5m to 3m.

(3) Frame loss rate: frame loss rates are below 6% as the distance increases from 0.5m to 3m.

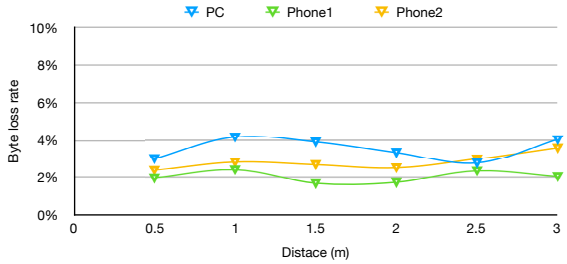
(4) Data loss rate: the data loss rate measured by iPerf2 are below 4% as the distance increases from 0.5m to 3m.

IV. Varying AP TX power

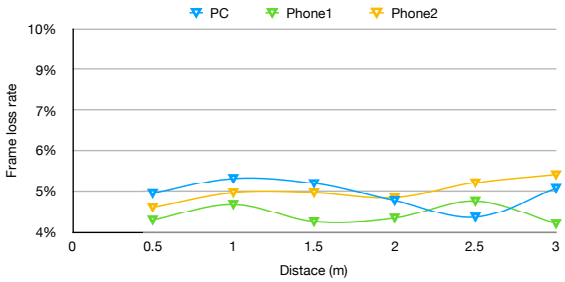
In this experiment, we seek to find out how the TX power affects performance, and what is the sensitive power range. We set the receiving distance to 5m, fix other parameters as default and vary the AP TX power. The results show that, there is a TX power range (around 510 dBm range) which contributes to throughput increase quasi-linearly. As the TX power further increases, it is no longer the throughput bottleneck. Other factors like receiving distance, network interferences and congestions may limit the maximum throughput.



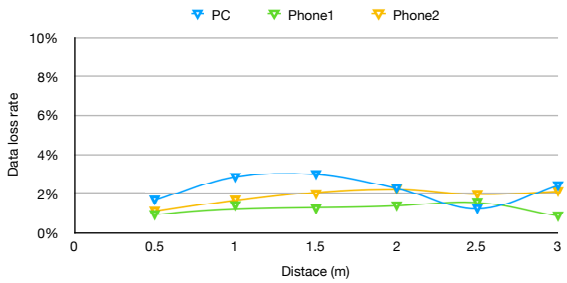
(a) UDP throughput of varying receiving distance



(b) Byte loss rate of varying receiving distance

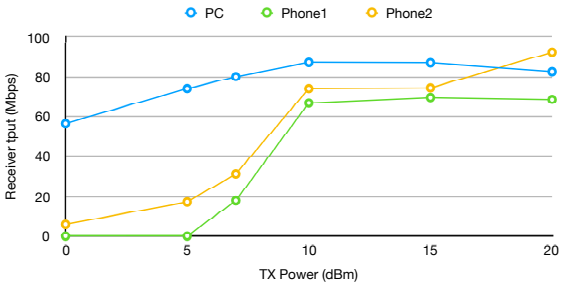


(c) Frame loss rate of varying receiving distance

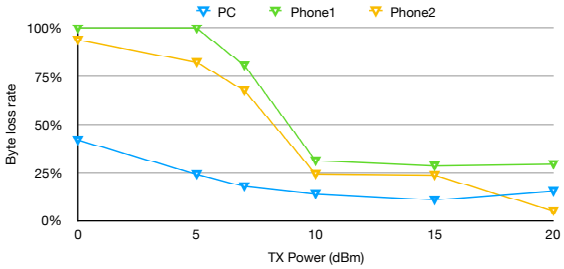


(d) Data loss rate of varying receiving distance

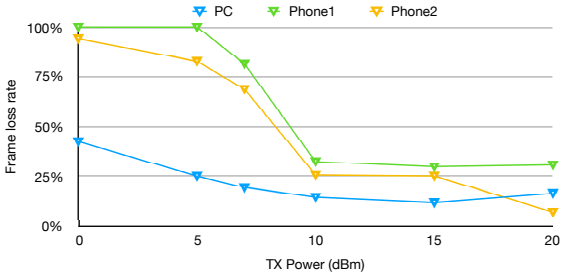
Figure 12: Varying receiving distance



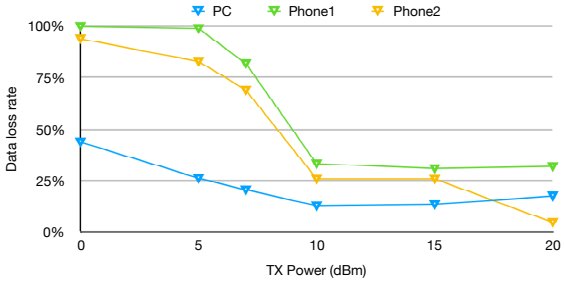
(a) UDP throughput of varying AP TX power



(b) Byte loss rate of varying AP TX power



(c) Frame loss rate of varying AP TX power



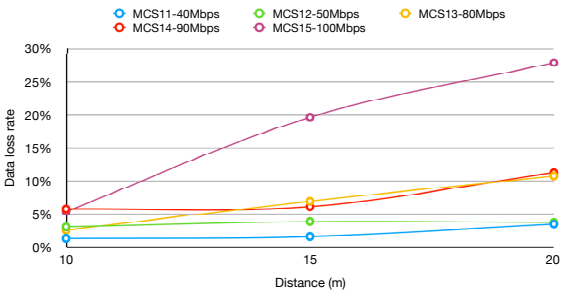
(d) Data loss rate of varying AP TX power

Figure 13: Varying AP TX power

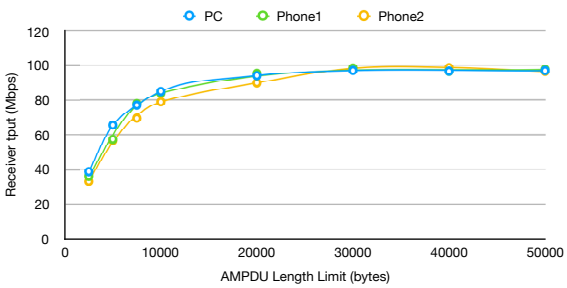
V. Varying MCS index and receiving distance

We next vary the MCS index and receiving distance while fixing other parameters with default values to test the limit of our multicast data transmission. We use the PC (Microsoft Surface tablet) as the receiver station since its receiving capacity is more stable.

Through the following experiments results, the lower MCS index will have lower throughput but lower packet loss rate at the same time. For some applications doesnt need the throughput as higher as 100 Mbps, lower MCS index can provide lower packet loss rate and support many applications at the same time.



(a) Data loss rate at varying distances and MCS indexes



(b) Receiver Throughput at varying A-MPDU length limit

Figure 14: Varying receiving distance and A-MPDU length limit

VI. Varying A-MPDU length limit

In this experiment, we test how A-MPDU length may impact the throughput. We change the A-MPDU length limit and fix other parameters as default. The maximum throughput increases

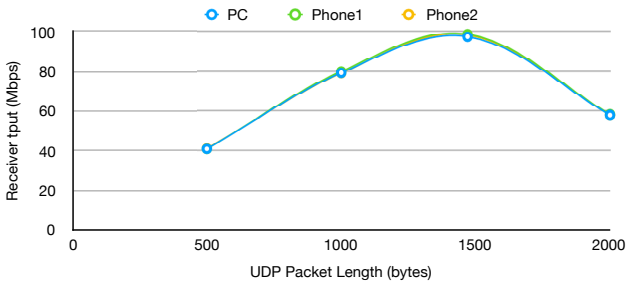


Figure 15: Receiver Throughput at varying UDP packet length

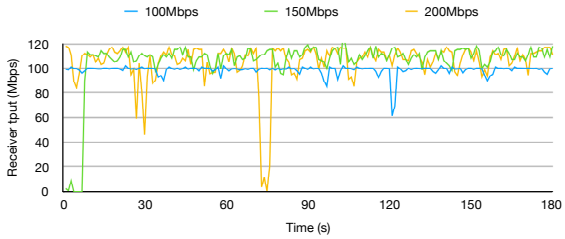


Figure 16: Receiver Throughput at varying MIMO settings

when the A-MPDU length threshold is larger. Because capacities of different devices of receiving A-MPDU are different, and when the packet loss happens, longer A-MPDU will cause longer aggregated frames lost at the same time. In our tests, the 30000 bytes A-MPDU length limit showed maximum throughput.

VII. Varying UDP packet length

In this experiment, we test how data packet length affects the achieved throughput. We fixed all parameters as default except varying the UDP packet length. The following results show that increasing UDP packet length yielded higher throughput and reached the peak at 1470 bytes. However, once the UDP packet length exceeds 1470 bytes, due to IP fragmentation (by default OS setting and MTU 1500 bytes), the throughput will be lower than the maximum value.

VIII. Changing MIMO settings

In this experiment, we test how using more antennas affects the multicast performance. We use a MacBook Pro 2017 as the station, which supports 3x3 MIMO. We set the multicast MCS index to 23 (216.7 Mbps PHY rate given Short GI and 3x3 MIMO), fix other parameters as default and vary the sending rate.

The achieved throughput is shown below. Over all, the maximum sustainable throughput achieved is up to 120 Mbps, while there were some fluctuations when the sending rate increases to 200 Mbps.

IX. Resilient Mechanism

By comparing the packet loss rate with and without the resilient mechanism, we can see that the retransmission can reduce the packet loss rate by more than 10 times on the phones. Even though the antenna of the tablet is more powerful than phones and it shows a lower packet loss rate without the resilient mechanism, we can still observe that the resilient mechanism can reduce the packet loss rate by 4 times. The results show that resilient Wi-Fi multicast can achieve high rate and low packet loss rate at the same time.

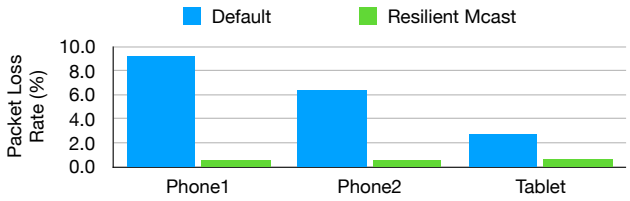


Figure 17: Resilient Mechanism Performance

7.4 Summary of the Experimental Findings

From our experiments, we can draw the following conclusions:

1. Our solution can yield the highest sustainable data throughput in multicast up to 110 Mbps running 802.11n at 5GHz (3x3 MIMO, Short GI, MCS 23, 5/6 coding rate, 64-QAM modulation).
2. The data throughput increases linearly as the sender rate increases until the maximum throughput is reached.
3. The byte loss percentage is at around 5% for throughput below or at the maximum.
4. On Android phones, due to processing overhead and congestion, increasing sending rate beyond the maximum throughput would make the data throughput unstable and loss rate surge.
5. The multicast data throughput at 20 meters can still sustain well above 50 Mbps, which should suffice for many realtime streaming applications. Using lower MCS under such scenarios would be better, as the stability improves and loss rate decreases.
6. It is safe to use the UDP packet size below 1470; the throughput will be larger if the packet size is longer.
7. Resilient multicast mechanism can reduce the packet loss rate for more than 10 times.

8 Discussion and Future Works

8.1 Issues & Next Step

We further identify two items to work on next:

1. The multicast is using the fixed rate that is manually configured. We will devise a rate adaptation algorithm that enables the multicast sender to adjust its transmission rate dynamically based on the channel conditions and clients' status.
2. Our current implementation for the link-layer multicast sender is at the AP side, while the smartphones act as the multicast receivers. We plan to prototype the smartphones so that they can be the multicast sender. This allows for more flexibilities with the AR applications on the smartphones.

8.2 Remaining Challenges

We now identify next-step issues in our wireless design and implementation. On the link-layer multicast component, we have two items to work on next:

1. The multicast is using the fixed rate that is manually configured. We will devise a rate adaptation algorithm that enables the multicast sender to adjust its transmission rate dynamically based on the channel conditions and clients' status.

2. Our current implementation for the link-layer multicast sender is at the AP side, while the smartphones act as the multicast receivers. We plan to prototype the smartphones so that they can be the multicast sender. This allows for more flexibilities with the AR/VR applications on the smartphones.

On the smart Wi-Fi with learning and reasoning, we plan to work on the following items:

1. The smartphone device should learn more knowledge about current state by exploiting more parameters and logs at the client side. We plan to incorporate more network analytics from the latest MobileInsight design for cross-layer information and sensory data at the smartphones. We further plan to enhance better reasoning capabilities by leveraging our recent work on using Bayesian network inference to infer the root causes for more soft failures.

2. At the AP side, we plan to further strengthen the reasoning function and the decision-making component. To this end, we plan to see how to leverage the crowdsourcing results from clients to make better decisions.

References

[1] Varun Gupta, Craig Gutterman, Yigal Bejerano, and Gil Zussman. Experimental evaluation of large scale wifi multicast rate control. *IEEE Transactions on Wireless Communications*, 17(4):2319–2332, 2018.

[2] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, pages 202–215, New York, NY, USA, 2016. ACM.

[3] Charles E. Perkins, Mike McBride, Dorothy Stanley, Warren "Ace" Kumari, and Juan-Carlos Ziga. Multicast Considerations over IEEE 802 Wireless Media. Internet-Draft draft-ietf-mboned-ieee802-mcast-problems-04, Internet Engineering Task Force, November 2018. Work in Progress.

[4] Yeonchul Shin, Munhwan Choi, Jonghoe Koo, Young-Doo Kim, Jong-Tae Ihm, and Sunghyun Choi. Empirical analysis of video multicast over wifi. In *Ubiquitous and Future Networks (ICUFN), 2011 Third International Conference on*, pages 381–386. IEEE, 2011.