WiFi and Multiple Interfaces: Adequate for Virtual Reality?

Huanle Zhang[*]
University of California, Davis
USA
Email: dtczhang@ucdavis.edu

Ahmed Elmokashfi
Simula Metropolitan Center for Digital Engineering
Norway
Email: ahmed@simula.no

Prasant Mohapatra
University of California, Davis
USA
Email: pmohapatra@ucdavis.edu

***Abstract*—In this paper, we investigate whether IEEE 802.11ac WiFi can support VR applications. To this end we conduct a controlled study of WiFi performance in an indoor setting. Our measurements reveal that WiFi transmissions suffer from high latency and jitter, which makes WiFi systems inadequate for VR applications. For example, the round-trip delay can be as high as $228ms$, and more than 24.2% packets experience jitter higher than $1ms$. To locate the root cause of the high latency and jitter, we dissect the network stack layer by layer and find that the main culprit is the wireless channel transmission time. To reduce the channel transmission time, we propose using multiple network interfaces running on non-overlapping channels. By using only two interfaces, we (1) reduce the median round-trip delay by 28.6% and jitters of higher than $1ms$ by 11.5% compared to the best single interface in UDP transmissions, and (2) reduce the median round-trip delay by 38.9% in TCP transmissions. We believe that this paper sheds some light on whether we can make today's WiFi systems VR-ready by using multiple interfaces.**

***Index Terms*—VR, 802.11ac, MPTCP, WiFi, Latency**

I. Introduction

Virtual Reality (VR) systems have undergone a rapid development in recent years. According to Business Insider, sales of VR headsets are expected to grow at a swift 99% compound annual growth rate between 2015 and 2020, creating an estimated annual revenue of $2.8 billion in 2020 only [1]. VR applications, however, pose unique challenges to network requirements in terms of bandwidth, latency and packet loss. Take $360°$ multi-perspective videos, which are one of the VR killer applications, as an example. They are data-hungry, latency-sensitive and also vulnerable to packet loss. A recent white paper by *Huawei* categorizes 360 VR video systems into four levels (see Table I) [2]. Several wireless technologies now are able to provide data rates that meet the expectations of all four levels (e.g., IEEE 802.11ad, 802.11ac).

TABLE I
Network requirements for 360 VR Videos From Huawei VR whitepaper [2]

Standard	Bandwidth	RTT	Packet Loss
Pre-VR	25 Mbit/s	40 ms	1.4E-4
Entry-Level VR	100 Mbit/s	30 ms	1.5E-5
Advanced VR	418 Mbit/s	20 ms	1.9E-6
Ultimate VR	2.35 Gbit/s	10 ms	5.5E-8

However, current VR products continue to rely on High-Definition Multimedia Interface (HDMI) cables. The cable not only limits the user's mobility and interferes with VR experience, but also creates a tripping hazard since the VR headset covers the user's eyes [3]. Currently, there are two commercial products that provide wireless connectivity for VR headsets [4]. These products use a proprietary wireless interface that operates in the $60GHz$ band. The high throughput that $60GHz$ band and mmWave in general provide makes them an obvious first choice for wireless VR [5]. There is, however, one key challenge, which is the vulnerability of mmWave to blockage of the line of sight between transmitters and receivers. Responding to this, Abari *et. al* built a system that uses reflectors to reroute signals around blockages [3]. While this solution can help mitigate blockages, it involves deploying extra components, i.e., reflectors. Further, reflectors placement is dependent on room geometry and layout. Others proposed hybrid WiFi and $60GHz$ systems [6], [7]. However, because of the high heterogeneous links, the overall performance can be worse than a single link [6]–[8], complicating system design and wasting networking resources.

WiFi systems operating in the $5GHz$ band and lower have witnessed huge success in the past decades. They are expected to continue to grow at a 17.8% compound annual growth rate between 2015 and 2020, creating $33.6 billion of market worth in 2020 only [9]. Therefore, leveraging WiFi transmissions to support VR applications benefits both WiFi and VR. The latest widely adopted WiFi standard, IEEE 802.11ac, comes with a theoretical speed up to $6.9Gbps$ [10]. Measurements of a large-scale enterprise WiFi system (Cisco Meraki) shows that 802.11ac clients are already running in *Gbps* data rates [11].

Despite the availability of high data rate WiFi networks, VR systems do not widely adopt WiFi transmissions. This paper takes a closer look at this lack of adoption and whether WiFi systems can be engineered to support VR. Our measurement reveals that WiFi transmissions suffer from high latency and jitter, which makes WiFi systems inadequate for VR applications. For example, we ran an experiment to measure the Round-Trip Delay Time (RTT) of two WiFi devices that are within one-hop direct communication link for a day (more than $800K$ data points collected). We find that in spite of 99.96% of the packets are echoed back within $10ms$, the RTT can be as high as $227.8ms$ and more than 24.2% packets

[*]Huanle Zhang interned at Simula Research Lab during this work

experience jitter higher than $1ms$. It is difficult to guarantee Quality of Service (QoS) for real-time multimedia applications in the presence of such high jitters [12], [13].

To locate the root cause of the observed latency, we dissect network stack layer by layer and measure the packets' stay time at each layer. Specifically, we record the timestamps of each packet entering UDP layer, IP layer, mac80211 layer, when the packet is passed to the WiFi driver for sending it out and the timestamp when the peer ACK is received by the driver. We find that the channel transmission time, i.e., the latency of a packet entering WiFi driver and being ACKed, dominates the latency.

To improve the channel transmission time, we propose to use multiple WiFi Network Interface Cards (NICs) running on non-overlapping channels. The reasoning is straightforward yet effective: the chances of accessing the wireless channel is higher when multiple interfaces are available than a single interface. To quantify the performance improvement, we prototype two devices each is equipped with two NICs. The experiment results of UDP transmissions show that we reduce the median RTT by 28.6% and jitters of higher than $1ms$ by 11.5% compared to the best single interface. We also evaluate the effectiveness of multiple-interface transmissions with MPTCP and the experiment results indicate that TCP packets witness even higher round-trip delay improvement than UDP, achieving 38.9% median delay reduction. We expect WiFi systems to fully support 360 VR videos by using more NICs.

In this paper, we make following contributions:

1) We measure the performance of WiFi systems in the context of VR applications. We observe that it is fluctuations in latency rather than bandwidth and packet loss that makes WiFi incapable for VR applications.
2) We locate the main culprit of the latency issue by dissecting the network stack layer by layer, and find that channel transmission time dominates the latency.
3) We quantify the performance improvement with regards to transmission time reduction by using multiple network interfaces. Our experiments cover both UDP and TCP.
4) We release our system codes, including kernel logging functionality, the MPTCP patch (v0.93) to OpenWrt, and data processing scripts at https://github.com/dtczhl/dtc-openwrt.

II. Background

In this section, we present a background of IEEE 802.11ac and multiple-interface based transmissions.

A. IEEE 802.11ac

IEEE 802.11ac is the latest WiFi standard that operates in the $5GHz$ band [10]. It supports wide bandwidth (up to $160MHz$), a large number of spatial streams (up to 8) and high-density modulation (e.g., 256-QAM).

We use the open source ath10k wireless driver for our 802.11ac WiFi modules. Ath10k is a WiFi driver for Qualcomm Atheros QCA98xx based 802.11ac devices. Regarding the driver as an independent layer in the network stack simplifies the study of WiFi performance in Linux (see Figure 1). In fact, the code interface between mac80211 and ath10k is similar to the interfaces between other layers (e.g., UDP to IP, IP to mac80211). Specifically, the upper layer interacts with the adjacent lower layer via the *struct of operations* that is implemented by the lower layer.

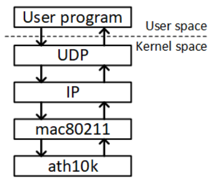


Fig. 1. Network stack from the perspective of Linux code structure

In the packet transmission path, the mac80211 layer passes a packet to the ath10k layer by calling the tx virtual function via the interface of ieee80211_ops ($.tx = ath10k\_mac\_op\_tx$). After obtaining the packet, the ath10k layer sends out the packet in a back-off manner (refer to [14] for the details of WiFi channel access). After receiving the acknowledgement from the peer, the ath10k layer reports the status of the packet to its upper layer via $ath10k\_txrx\_tx\_unref$ function. As a result, ath10k provides a couple useful information about packet transmission: (1) the timestamp at which a packet enters the ath10k layer for transmission (2) a status report for each transmitted packet.

The reception path of ath10k is based on NAPI (stands for New API), which is designed for high-speed networking transmission. Compared to previous purely interrupt based packet reception which creates thousands of interrupts per second in a high-speed transmission, NAPI combines interrupt and polling to effectively decrease system loads. Ath10k delivers the packet to the upper layer by calling $ath10k\_htt\_rx\_handle\_amsdu$ function.

To the best of our knowledge, existing schedulers do not consider these fine-grained information from the ath10k layer to schedule packets among multiple networking interfaces. As we show in Section V, the transmission-response delay of the ath10k layer strongly correlates with latency, and thus ignoring such information impairs the effectiveness of schedulers and VR user experience.

B. Multiple Network Interfaces

An increasing number of devices have multiple network interfaces [15]. For example, smartphones have cellular and WiFi interfaces, and datacenter servers have multiple Ethernet interfaces. This availability has motivated the use of multiple network interfaces to increase throughput, reduce latency and improve robustness. One of the most successful multiple-interface transport protocol is Multipath TCP (MPTCP) [16], which has been standardized at the IETF. Many MPTCP schedulers have been proposed based on, e.g., lowest RTT first (MinRTT), bulk data download time (DEMS [17]) and fairness to other traffic flows [18].

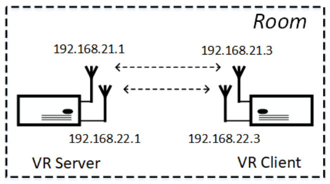


Fig. 2. Networking setup

In this paper, we compare the performance of multiple-interface based transmissions to the single-interface counterpart. Different from existing systems with heterogeneous interfaces, e.g., one cellular interface and one WiFi interface, we focus on interfaces of same kind to study WiFi systems. Specifically, each device in our setup is equipped with two WiFi NICs of same model, both running 802.11ac.

III. Methodology and Setup

This section gives details about our measurement and data collection setup.

A. Devices

We use two PC Engines apu2 boards [19] to run experiments. An apu2 board has two MiniPCI interfaces. We use them for 802.11ac modules. We attach a $64G$ SSD memory to the mSata interface. An apu2 board has four $1GHz$ CPU cores.

We install OpenWrt Operating System (OS) in the boards. The OS that we use is based on Linux kernel 4.9.82, with the wireless networking package of backports-2017-11-01. We transplant MPTCP (v0.93) for the experiments with TCP over multiple interfaces. Figure 2 shows the networking setup of the two boards. We emulate one board as the VR server and the other one as the VR client. Each device has two Qualcomm Atheros QCA9888 802.11ac NICs (model: WLE650V5-18A).

B. Environment

Our experiments are conducted in a research lab at UC Davis. The size of the lab is $\sim$6×10 meters. About five and two Ph.D. students/candidates work in the lab during weekdays and weekends respectively. We place the VR server and the VR client $\sim 6m$ apart, without line-of-sight, but within one-hop direct communication (i.e., no routing to other devices).

We fix the channel bandwidth of NICs to $80MHz$ during our measurement. Unless otherwise mentioned, the two interface pairs operate on non-overlapping channels: channel 36 ($5.180GHz$) and channel 149 ($5.745GHz$). To roughly characterize the background interference, we detect the surrounding WiFi SSIDs and find about 16 SSIDs in the $5GHz$ band, in which zero SSID in the channel 36 and three SSIDs in the channel 149.

C. Data logging

We add a 4-byte sequence number in the beginning of packet payloads for packet identification. We record timestamps of packets that are sent and received on the application layer, and timestamps that packets entering and leaving each network layer. However, the network stack does not have file system support, which means we cannot log data from network stack directly to files. To circumvent this data logging problem, we use a memory based virtual file system called *debugfs* to save data from network stack to memory and then format the memory to files.

D. System Tuning

We fine-tune our system to support high throughput, and then we conduct experiments on RTT and packet loss. We set Type of Service (ToS) of IP packets to IP-TOS_THROUGHPUT. Both send and receive buffer sizes are set to the maximum for avoiding packet loss. Unless otherwise mentioned, other system configurations, e.g., maximum retry times of wireless driver and the adaptive transmission power, are remained to the defaults.

IV. WiFi and VR requirements

VR applications pose unique challenges to networks with regard to bandwidth, latency and packet loss. In this section, we empirically examine whether IEEE 802.11ac supports these requirements.

A. Throughput

Once the bandwidth, Modulation and Coding Scheme (MCS), Number of Spatial Streams (NSS) and Guard Interval (GI) are determined, the data rate of an 802.11ac device can be found by looking up the data rate table (e.g., in [10]). For example, we use WLE650V5-18A 802.11ac modules. This WiFi module supports $80MHz$ bandwidth, two spatial streams ($NSS = 2$), 256-QAM ($MCS = 9$) and short GI, and thus it can provide up to $866.7Mb/s$ data rate.

The ath10k driver dynamically adjusts interface parameters according to the channel conditions. It uses large MCS and NSS for clean channels, and vice versa. We notice that it occasionally changes bandwidth for a short duration as well, from $80MHz$ to $40MHz$ or $20MHz$. The driver stays on short GI for our 802.11ac modules as far as we observe. During our measurements, the ath10k driver reports data rate of $195.0Mb/s$ to $866.7Mb/s$, with MCS of 4 to 9, and NSS of 1 to 2. The mean data rate reported is around $500Mb/s$.

Since 802.11ac has already supported extremely high data rate, the VR throughput requirement is not a reason of not adopting WiFi transmissions. Specifically, the 802.11ac standard allows up to $6.9Gbps$ data rate [10]; the measurement of a large-scale enterprise WiFi system (Cisco Meraki) shows that some 802.11ac clients are already running in $Gbps$ data rates [11], which indicates that existing commercial WiFi networks have already supported the throughput requirement of Advanced-level VR.

B. Packet Loss

We fix the packet sending interval to $100ms$ and collect more than 22 million UDP data points for 23 days in a row, but no packet loss is observed. This may seem counterintuitive since we use unreliable UDP transport layer for this experiment. However, in our setting, the server and the client communicate via one-hop link, and the default system configurations (e.g., retry times of the wireless driver) successfully delivers packets to its peer device.

C. Latency

To measure RTTs, we record two timestamps in the VR server when a packet enters the UDP layer and when the packet is received by the UDP layer. Please note that the RTTs measured on the transport layer are smaller than the RTTs from the application layer. We collect more than 20 million data points. Although the median RTT is about $1.9ms$ for all these days, the maximum delay is as high as $236ms$, which is intolerable for VR applications.

Considering that existing WiFi systems satisfy throughput and packet loss of VR requirements, we focus on latency analysis in the rest of this paper.

V. Latency Analysis: Single Interface

In this section, we analyze the latency when using a single 802.11ac interface. Specifically, we only enable the $192.168.21.0/24$ subnet (refer to Figure 2).

Figure 3 depicts the RTTs from a one-day trace. The trace contains 862650 UDP packets. We can see that RTT varies significantly and some packets suffer very high delays. It also illustrates the effect of background traffic on RTTs, as packets from $13:00PM$ to $17:30PM$ tend to have higher delays; there are more people in the research lab during these hours.

We plot the CDF of the RTT trace in Figure 4. We can see that half of the packets have an RTT between $0.7ms$ and $1.8ms$. However, 0.04% packets have RTTs higher than $10ms$ with the maximum delay of $227.8ms$. These high-delay packets lead to poor user experience for VR applications. Figure 5 shows the complementary CDF of the absolute jitter. To ensure QoS for real-time multimedia applications, the delay variation, i.e., jitter, should be less than $1ms$ [12], [13]. The jitter distribution is spread over several orders of magnitude. Although half of the packets have jitters less than $0.5ms$, there are 24.2% packets having jitter higher than $1ms$ and more than 1% packets having jitters greater than $3.8ms$, with the maximum jitter of $226.9ms$. The high delay and the high jitter imply that WiFi systems cannot consistently guarantee the QoS needed by real-time high data-rate VR applications.

To locate the root cause of the observed extreme high latencies, we dissect the network stack layer by layer and measure the packets' stay time at each layer. Specifically, we record for each packet, the time as it enters the UDP layer, IP layer, mac80211 layer, and also when it is passed to the WiFi driver for sending out and when the ACK is received from the WiFi driver of the peer.

A. Delays from Upper Layers

Figure 6 shows the delays of packets going down from the UDP layer to the IP layer. The delay is negligible, with the minimum, median and maximum of $10\mu s$, $16\mu s$ and $86\mu s$, respectively. Figure 7 shows the same for the packets going from the IP layer to the mac80211 layer. The minimum, median and maximum delay are $9\mu s$, $16\mu s$ and $119\mu s$, respectively. We plot the delays of packets going down from mac80211 layer to ath10k driver in Figure 8. The processing is even more lightweight compared to other layers. The minimum, median and maximum are $5\mu s$, $9\mu s$ and $48\mu s$ respectively. Considering all the processing delays from upper layers, the minimum, median and maximum delays are $23.9\mu s$, $41.0\mu s$ and $145.0\mu s$, respectively.

B. Delays from Channel Transmission

We record the time difference between the timestamps that a packet enters the ath10k driver layer until the driver receives the acknowledgment from the WiFi driver of the peer. We regard this time difference as channel transmission time because it represents the time length that the driver needs to deliver the packet to its peer over the wireless channel. Figure 9 shows the channel transmission time. We have the following observations: (1) The channel transmission time dominates the RTT. The minimum, median, maximum delay of the channel transmission are $0.3ms$, $0.9ms$ and $127ms$, respectively. (2) As the background traffic increases, so are the channel transmission delays. We can see that from $13:00PM$ to $17:00PM$ when more people are working in the building, the delays tend to be higher. (3) Comparing Figure 3 and Figure 9, we can see that they have similar envelope. The correlation coefficient between them is 0.71, which means that the channel transmission time is a good indicator of the RTT. We further plot the CDF of the channel transmission time in Figure 10 and have an additional observation: (4) The channel transmission time follows multiple uniform distribution. We manually add inflection points to segment the CDF curve. Each curve segment exhibits very good linearity. This observation is consistent with the behavior of the exponential-based backoff algorithm.

C. Packet Analysis on Receiver's Side

Since the channel transmission time is correlated with RTTs, we expect it to correlate with the packet reception as well. We plot the jitter in packet reception in Figure 11. The jitters are mostly symmetric along the y-axis in that the jitters from two adjacent packets tend to cancel each other. Comparing Figure 11 with Figure 9, we can see that they have similar envelopes. The correlation coefficient between them is 0.62. Therefore, channel transmission time is also a good indicator for the jitters of packet reception.

D. Summary of Single-Interface WiFi Transmission

We highlight some findings from the measurement analysis on single-interface WiFi transmissions.

1) Single-interface based WiFi transmissions cannot adequately support VR applications because of the outlier RTTs (as high as $227.8ms$) and the jitters (24.2% packets have jitter higher than $1ms$).

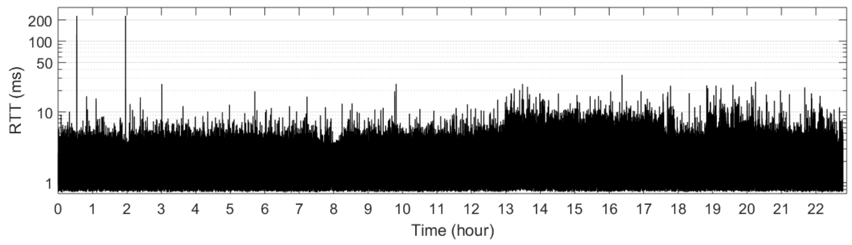


Fig. 3. One day trace of RTT using single interface

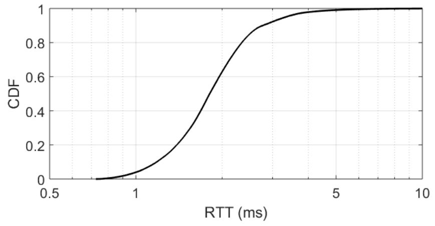


Fig. 4. CDF of the RTT using single interface

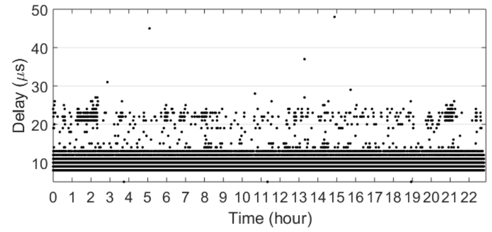


Fig. 8. Delays of packets going down from mac80211 layer to ath10k driver

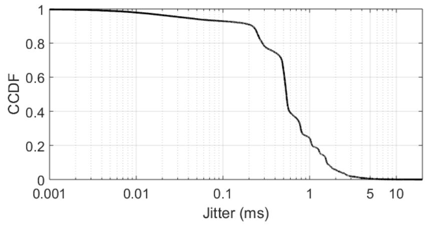


Fig. 5. Complementary CDF of the jitter using single interface

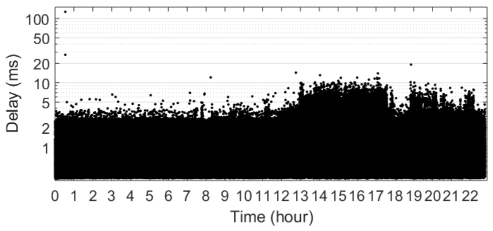


Fig. 9. Channel transmission time using single interface

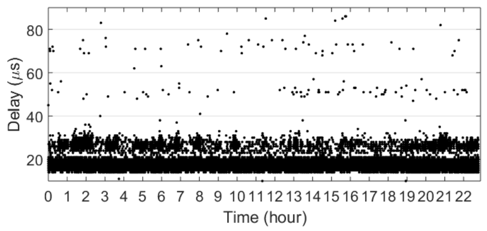


Fig. 6. Delays of packets going down from UDP layer to IP layer

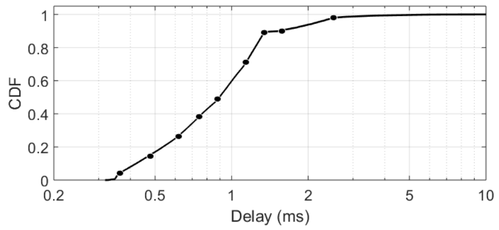


Fig. 10. CDF of channel transmission time

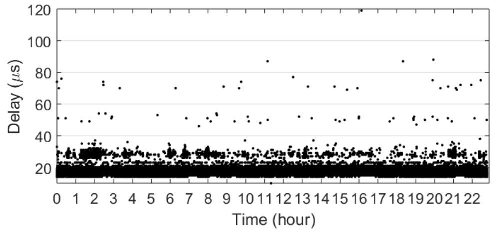


Fig. 7. Delays of packets going down from IP layer to mac80211 layer

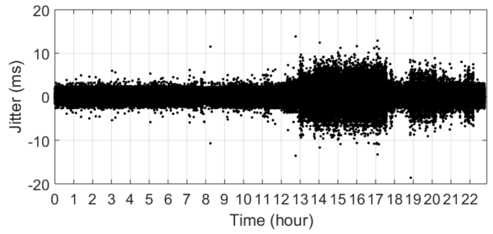


Fig. 11. Packet reception jitter on the receiver side

2) The channel transmission time accounts for the high and variant RTTs. Compared to upper layers combined, delays from channel transmission time is about 22 times in the median delay case and 876 times in the maximum delay case.
3) The channel transmission time is closely correlated

with the RTTs (correlation coefficient: 0.71) and the packet reception jitters on receivers (correlation coefficient: 0.62). Therefore, incorporating the information of channel transmission time in packet schedulers have the potential to improve efficiency, especially for packet scheduling among multiple interfaces.

VI. Latency Analysis: Multiple Interfaces

To reduce the channel transmission time of packets, we propose using multiple interfaces running on non-overlapping channels. The reasoning is straightforward yet effective: the chances of accessing the wireless channel is higher when multiple interfaces are available than a single interface.

For the experiments with multiple interfaces, we enable both subnets (refer to Figure 2). We consider the simple yet optimal packet scheduling strategy: each packet is duplicated to both interfaces. This scheduling strategy is optimal with regard to packet latency reduction because the receiver receives the earlier packet between the two interfaces. We record timestamps on user-space, and disable the kernel logging functionality. Our main focus is on the performance improvement of multiple interfaces compared to a single interface.

A. UDP Improvement

We analyze a one-day trace that includes more than $800K$ UDP data points.

1) Round-Trip Delays Improvement: Figure 12 shows the RTTs for interface 1, interface 2 and the combined interface. Because interface 1 and interface 2 run on non-overlapping channels, their RTTs are expected not to be strongly correlated. This is confirmed by the bottom figure in which the combined interface has much smaller and smoother RTTs than individual interfaces.

Figure 13 plots the complementary CDFs for the RTTs. The median RTT is $2.8ms$ and $3.0ms$ for interface 1 and 2 respectively. In comparison, the combined interface reduces the median RTT to $2.0ms$, achieving 28.6% RTT reduction compared to the best single interface (i.e., interface 1). In addition, the combined interface has shorter tail than individual interfaces. For example, considering the worst 90% to 99% packets, the combined interface has an RTT range of $2.8ms$ ($3.8ms - 6.6ms$), whereas interface 1 and 2 spans for $3.6ms$ ($4.9ms - 8.5ms$) and $11.2ms$ ($6.0ms - 17.2ms$), respectively.

2) Jitter Improvement: Figure 14 plots the complementary CDFs for the jitter when the packets are sent back to the sender. The combined interface effectively reduces the percentages of jitters higher than $1ms$ by 11.5% and 18.11% compare to interface 1 and interface 2, respectively. The jitter is worse than the data shown in Figure 5, because we collect data on application layer in multiple-interface experiments.

Figure 15 shows the one-way jitter from the sender to the receiver. The narrower the PDF is, the less variation of the packet reception. As we can see, the combined interface is centered around $100ms$ which is the packet sending intervals. Take the best 90% packets for example: the width for the combined interface is $1.84ms$, whereas it is $4.31ms$ and $7.38ms$ for interface 1 and interface 2, respectively.

B. TCP Improvement

We transplant MPTCP (v0.93) into OpenWrt and conduct experiments to explore the performance gain of TCP communication using multiple homogeneous WiFi interfaces versus the single-interface counterpart. We set the scheduler of MPTCP to *redundant* in which data are sent on all available subflows and thus results in lowest possible latency. More than $100K$ TCP data points are collected.

Figure 16 depicts the median RTTs, and the 5% and 95% percentiles. The median RTT is improved from $1.8ms$ to $1.1ms$, achieving 38.9% RTT reduction. It is interesting to note that TCP witnesses a higher RTT reduction than UDP (28.6%). Using multiple interfaces for TCP communications also greatly reduces packets that suffer from high RTT. The 95% percentiles are improved from $4.1ms$ in single-interface case to $1.9ms$ in multiple-interface case, achieving as large as 53.7% reduction. The 5% percentile is $1.0ms$ and $0.8ms$ respectively, which achieves 20.0% improvement in multiple-interface data transmission. Compared to UDP, TCP benefits more from multiple-interface transmission.

C. Channel Correlations

The benefits of using multiple interfaces rely on the used channels being independent so that the combined interface has smaller and smoother RTTs. We change the channels of the two NICs to measure the correlation between channels. We also change the packet transmission interval from $100ms$ to $10ms$ and collect 10-minute's data for each channel pair. This is important for capturing short-lived correlations. In the US, the unlicensed $5G$ band supports six $80MHz$ channels [11]. Figure 17 shows the channel correlation for two experiment runs. The interfaces fail to work when both operate in same channels (i.e., diagonal numbers in the figure), and we put 1 for illustration purposes. We can see that different channels have different correlations. Even for the same channel pairs, the correlation changes for different days. The irregularity indicates that channel selection for multiple interfaces is not as simple as it seems.

D. Summary of Multiple-Interface WiFi Transmission

Multiple-interface based WiFi transmissions effectively reduce RTTs and jitters. To benchmark the performance improvement of multiple-interface transmission, we duplicate packets among all available flows/channels. From our experiments in which each device has two interfaces, the combined interface reduces the RTT of UDP packets by 28.6% and the jitters of higher than $1ms$ by 11.5% compared to the best single interface. The experiments with TCP show that TCP packets witness even higher round-trip delay improvement than UDP, achieving 38.9% median delay reduction. Although the combined interface still has a few high-delay packets, the number of outlier packets is largely reduced. We anticipate that by incorporating more interfaces (e.g., 3 or 4) and adaptive multipath algorithms, the WiFi system could further reduce RTTs and jitters and thus fully support VR applications.

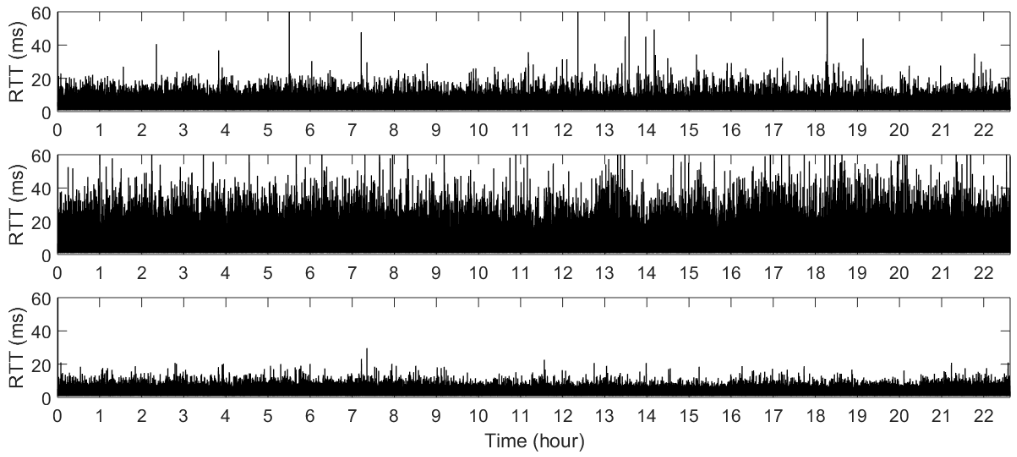


Fig. 12. Round-trip delays for multiple interfaces. Top to bottom: interface 1, interface 2 and the combined interface

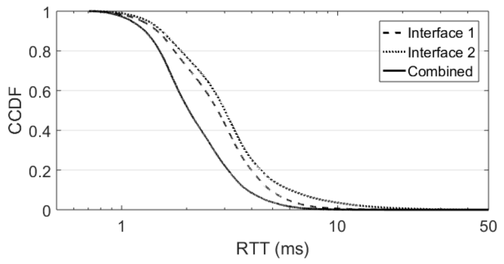


Fig. 13. CCDF for the UDP round-trip delays using multiple interfaces

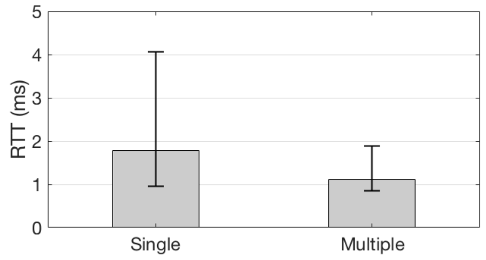


Fig. 16. The median, 5% and 95% percentiles of RTTs for single-interface and multiple-interface TCP packet transmissions

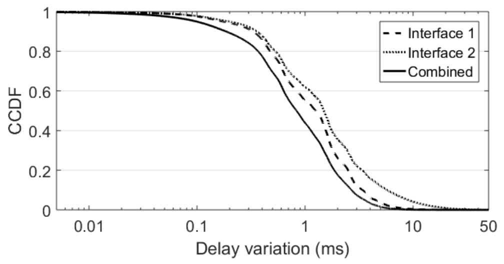
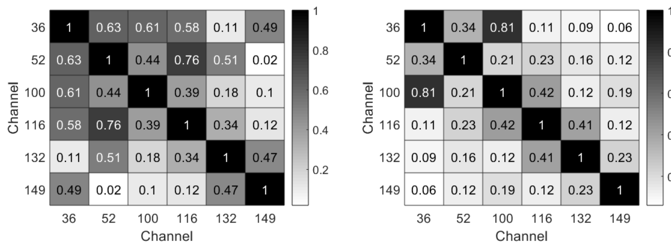


Fig. 14. CCDF for the UDP jitters using multiple interfaces



(a) Experiment one (b) Expriment two

Fig. 17. Channel correlations

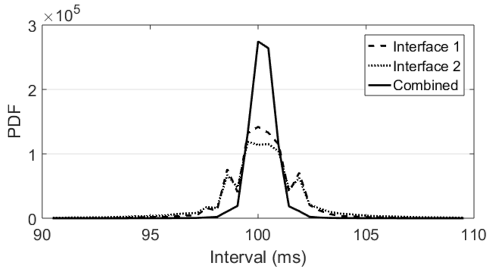


Fig. 15. PDF for UDP jitters on the receiver using multiple interfaces

VII. Related Work

We cover the related work with regards to the VR transmission systems and WiFi measurements.

A. VR Transmission Support

Current VR devices rely on wire-line transmissions which have inherent disadvantages, such as terrible user experience and potential safety issues. $60GHz$ techniques (e.g., IEEE 802.11ad) are promising to provide wireless transmissions of high throughput and low latency. However, due to the narrow beams and high signal attenuation, $60GHz$ cannot perform well during mobility and blockage [6], [20] which are unavoidable for VR interactions. Abari *et. al* built a system that uses reflectors to reroute around blockages [3]. While this solution can help mitigate blockages, it involves deploying extra components, i.e., reflectors. Further, reflectors placement is dependent on room geometry and layout. Hybrid systems have been proposed to leverage multiple network interfaces on devices [6], [7], [21]. However, because of the high heterogeneous links, the overall performance can be worse than a single link [6]–[8], complicating system designs and wasting networking resources. VR support in LTE networks is studied in [22]. This paper takes a closer look at whether pure WiFi systems can be engineered to support VR applications.

B. WiFi Measurements

There are a plethora of works on WiFi performance analysis, including system measuring [11], [23], pathology detection [14] and interference deconstruction [24]. Most of existing works on studying commercial WiFi systems either rely on higher-layer system logging data [14] or closely-placed WiFi NICs as sniffers [25]. This paper substitutes to existing WiFi measurement works in that we provide fine-grain analysis (by directly dissecting network stack) of the latest WiFi standard (i.e., IEEE 802.11ac) and also we explore multiple network interfaces to mitigate the latency issue of pure WiFi transmissions.

VIII. Future Work

The long-term goal of our project is to build a purely WiFi based transmission systems for VR applications, including but not limited to 360 VR videos. We believe that using multiple NICs is the right direction for building extremely high throughput, low latency and robust WiFi networks.

More WiFi NICs. We plan to explore the performance limit that a WiFi system can provide by using a large number of WiFi NICs running on non-overlapping channels. In the US, for example, the unlicensed $5G$ band supports six $80MHz$ channels [11]. Therefore, a WiFi system can use six $80MHz$-NICs without interfering to each other.

Multiple Interface Scheduling. In this paper, we duplicate packets among all available channels to explore the performance limit of purely WiFi based VR solutions. In practice, the benefits of using multiple WiFi NICs cannot be fully leveraged without an effective multiple-interface packet scheduler. Existing schedulers are designed for general network interfaces, which are not optimal for purely WiFi systems since they do not consider the variation of wireless channel transmission delays.

IX. Conclusion

In this paper, we investigate whether purely Wi-Fi based transmission systems can support VR applications. By preliminary measurements, we find that high delays and jitters make single-interface WiFi inadequate. By dissecting network stack layer by layer, we locate the main culprit, i.e., wireless channel transmission time. Based on the findings, we propose to use multiple interfaces to reduce channel transmission time. To explore performance limit of multiple interface transmission, we duplicate packets among all available channels. The measurement results show that our multiple-interface system is effective in mitigating the latency. We believe that using multiple interfaces is the right direction to make WiFi systems fully support VR applications.

References

[1] Andrew Meola. Facebook plans to ship a mind-blowing number of oculus rift units in the next two years. Technical report, Business Insider, 2016. http://www.businessinsider.com/facebook-expects-to-ship-26-million-oculus-rifts-by-2017-2016-4.

[2] Huawei. Whitepaper on the VR-oriented bearer network requirement (2016). Technical report, Huawei Technologies, 2016.

[3] Omid Abari, Dinesh Bharadia, Austin Duffield, and Dina Katabi. Enabling high-quality untethered virtual reality. In *USENIX NSDI*, 2017.

[4] Tpcast. https://www.tpcastvr.com/product. Accessed: 2018.

[5] Luyang Liu, Ruiguang Zhong, Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. Cutting the cord: Designing a high-quality untethered VR system with low latency remote rendering. In *ACM MobiSys*, 2018.

[6] Sanjib Sur, Loannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. Wifi-assisted 60 ghz wireless networks. In *ACM MobiCom*, 2017.

[7] Swetank Kumar Saha, Roshan Shyamsunder, Naveen Muralidhar Prakash, Hany Assasa, Adrian Loch Dimitrios Koutsonikolas, and Joerg Widmer. Poster: Can mptcp improve performance for dual-band 60 ghz/5 ghz clients. In *ACM MobiCom*, 2017.

[8] Yung-Chin Chen, Yeon sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. A measurement-based study of multipath TCP performance over wireless networks. In *ACM IMC*, 2013.

[9] Marketsandmakets. Gobal wi-fi market worth 33.6 billion usd by 2020. Technical report, 2015. https://www.marketsandmarkets.com/PressReleases/global-wi-fi.asp.

[10] 802.11ac-2013 part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications. Standard, IEEE, 2013.

[11] Apurv Bhartia, Bo Chen, Feng Wang, Derrick Pallas, Raluca Musaloiu-E, Ted Tsung-Te Lai, and Hao Ma. Measurement-based, practical techniques to improve 802.11ac performance. In *ACM IMC*, 2017.

[12] Hongqiang Zhai, Xiang Chen, and Yuguang Fang. How well can the ieee 802.11 wireless lan support quality of service. *IEEE Transactions on Wireless Communications*, 4(6):1536–1276, 2005.

[13] ITU G.1010. End-user multimedia qos categories. Technical report, ITU, 2001.

[14] Partha Kanuparthy, Constantine Dovrolis, Konstantina Papagiannaki, Srinivasan Seshan, and Peter Steenkiste. Can user-level probing detect and diagnose common home-WLAN pathologies? *ACM SIGCOMM Computer Communication Review*, 42(1):7–15, 2012.

[15] Juan Antonio Cordero. Multi-path tcp performance evaluation in dual-homed (wired/wireless) devices. *Journal of Network and Computer Applications*, 70:131–139, 2016.

[16] Mptcp. https://www.multipath-tcp.org. Accessed: 2018.

[17] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. Accelerating multipath transport through balanced subflow completion. In *ACM MobiCom*, 2017.

[18] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *USENIX NSDI*, 2011.

[19] Pc engines apu2 platform. https://www.pcengines.ch/apu2.htm. Accessed: 2018.

[20] Teng Wei and Xinyu Zhang. Pose information assisted 60 ghz networks: Towards seamless coverage and mobility support. In *ACM MobiCom*, 2017.

[21] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z. Morley Mao, and Subhabrata Sen. An in-depth understanding of multipath tcp on mobile devices: Measuring and system design. In *ACM MobiCom*, 2016.

[22] Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. Supporting mobile VR in LTE networks: How close are we? In *ACM SIGMETRICS*, 2018.

[23] Domenico Giustiniano, David Malone, Douglas J. Leith, and Konstantina Papagiannaki. Measuring transmission opportunities in 802.11 links. *IEEE/ACM Transactions on Networking*, 18:1516–1529, 2010.

[24] Shravan Rayanchu, Ashish Patro, and Suman Banerjee. Catching whales and minnows using wifinet: Deconstructing non-wifi interference using wifi hardware. In *USENIX NSDI*, 2012.

[25] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. Mojo: A distributed physical layer anomaly detection system for 802.11 WLANs. In *ACM MobiSys*, 2006.