UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

TIME AND FREQUENCY CHARACTERISTIC OF 802.11AX

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

Master of Science in Electrical and Electronic Engineering

By

JIAMIAO ZHAO
Norman, Oklahoma
2019

TIME AND FREQUENCY CHARACTERISTIC OF 802.11AX

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Hazem Refai, Chair

Dr. Thordur Runolfsson

Dr. Samuel Cheng

Table of Contents

List of Tables

List of Figures

Abstract

Wireless Local Area Network (WLAN) is a wireless communication system that connects two or more devices using radio frequency to form a local network (LAN). WLAN is used inside buildings (e.g., apartment, campus, train station). Unlike Ethernet, WLAN users remain connected while in motion. The first WLAN standard, namely IEEE 802.11b, was released in the 1990s. Maximum achievable data rate is 11 Mbps for 802.11b. The most recent version is 802.11ax with maximum throughput of 3.46 Gbps. This thesis reports network throughput, jitter, and delay performance of IEEE 802.11ax experimentally measured and analyzed under various conditions. The investigation attempted to build a relationship between the network performance given a particular factor, such as Signal to Noise Ratio (SNR), packet size, and communication protocol. This work also discussed and examined the way in which the Transmission Control Protocol (TCP) mechanism and TCP congestion control affected throughput.

The empirical data showed that 802.11ax throughput is function of SNR. Hence, models were developed using SNR as input and throughput as output. The models are able to predict throughput according to SNR. Throughput was shown to be influenced by packet size, window size, and packet loss.

Chapter 1 Introduction

1.1 Introduction

Since IEEE 802.11 was established at the end of the 20th century, the protocol has been improved to enhance throughput and accommodate new technologies. These modifications improve 802.11ac standard throughput to achieve 3.46Gbps by introducing new modulation and coding schemes and building Multi-User Multiple Input Multiple Output (MU-MIMO).

In 2018, Wi-Fi facilitated more than 50% of all internet traffic. By 2020, over 35 billion Wi-Fi devices are expected to flood the market. The cumulative shipments of IoT wireless devices adds to the daily excitement of WLAN. With the exception of improving data rate at the physical layer, the new Wi-Fi standard is expected to increase technological capacity and provide a seamless connection among networked devices. In 2019, IEEE 802.11ax was released to overcome anticipated challenges. This thesis was designed to provide empirical models for this standard. Doing so is important because WLAN is extensively deployed in business-centered, apartment, and campus buildings.

Internet Service Providers (ISP) facilitating user access to the Internet are primarily charged with providing and maintaining an Internet connection, whether the user is at work or at home. WLAN also provides voice and steam services. Therefore, network throughput and delay must be analyzed before installing the internet access point. In this thesis, several factors influencing 802.11ax throughput and delay performance were empirically measured and discussed. SNR, traffic control, packet size, and packet loss, as well as the way in which theses parameters affect

network performance, were analyzed. Empirical models were then derived to predict WLAN performance based on a receiver surrounding scenario. The model was also shown to assist engineers in optimizing single user network performance.

1.2 Preceding Research

Aruba [1] detailed the background of 802.11ax and briefly discussed new technologies implemented in the standard, as well as methods to improve overall network performance. IEEE 802,11ax was shown to improve maximum throughput four times over the previous 802.11ac standard. Researchers also introduced Orthogonal Frequency Division Multiple Access (OFDMA), which are used in Long-Term Evolution (LTE) and Multi-users Multiple Input and Multiple Output (MU-MIMO). Qiao [2] analyzed how OFDMA coordinated access point and devices so that the packet can simultaneously transmit or receive. These researchers developed a Medium Access Control (MAC) protocol, including new physical layer sensing, fast back off process, enhanced RTS/CTS mechanism, and frame structures. Qiao [2] also developed a mathematical model to predict maximum throughput by leveraging the new MAC protocol. Results proved that maximum throughput increased 160%. Jorden [3] developed a new channel access method for 802.11ax to accommodate OFDMA in WLAN. Researchers speculated that choosing multiusers in the PHY layer creates a new challenge, primarily because legacy Carrier Sense Multi Access/ Collision Avoidance (CSMA/CA) was designed for a single user at any given time. Researchers proposed a pragmatic and efficient OFDMA-based hybrid channel access (OHCA) method for 802.11ax, which

improved performance over the existing solution by lowering collision rate, shortening access delay, and facilitating fair bandwidth sharing (Jorden 2018).

Arjun [4] measured expected throughput performance by using spatial reuse of 802.11ax and determined that spatial reuse is a factor for improving throughput in a dense deployment scenario.

To allocate bandwidth in heterogeneous scenario created by different packet length, 802.11ax OFDMA, OFDMA divides one channel into multiple sub-channels. Notably, a 20MHz channel can be divided into no more than nine sub-channels. When a variety of clients simultaneously transmit packets to access point (AP), the technology packs frames from different clients into various sub-channels, and then simultaneously transmits those in a single channel. The previous standard used CSMA/CA (i.e., “listen before talk” scheme). This method ensured that each frame was transmitted across the full channel so that clients do not waste time on a contention and back-off window. It is important also to note that no more than one client is able to attempt communication with AP. Besides OFDMA, 802.11ax also implements a higher modulation and coding scheme (MCS), which increases maximum data rate under good conditions (i.e., received signal strength is higher than -55dBm). With these new technologies, 802.11ax maximum throughput is four times that of the 802.11ac protocol.

A Markov chain model was used in [5] to estimate energy efficiency when increasing the contention window for 802.11ax. Researchers also compared energy efficiency relative to the number of spatial steams for transmitting frames. This study reported that using MU-MIMO optimized efficiency with four bandwidths

offered by 802.11ax. [6] researchers examined the 802.11ax key technologies and enhancements. This work stated that 802.11ax has the potential to “work in Internet of Things (IoT) if WLAN overcomes the energy consumption, range and efficiency problems (due to diversity of loads)” (M. Shahwaize 2017). IEEE formed a new research group to overcome the power consumption and transmission range of WLAN in various scenarios.

1.3 Thesis Summary

This thesis focused on application layer performance (e.g., throughput, delay and packet loss) of 802.11ax under various conditions and also developed empirical models to predict network performance for future deployment. Testing was conducted on the fourth floor of Building Four at the University of Oklahoma-Tulsa. The remainder of this thesis is organized as follows:

Chapter 2 introduces 802.11ax and details its key enhancements. This information affords readers basic knowledge about the differences between 802.11ax and its previous version, namely 802.11ac. Chapter 3 discusses the experimental setup and configuration. The access point of 802.11ax is ASUS RT-AX88u, which operates on 2.4G and 5G band, and the router has four antennas. Iperf3, mrt and wireshark software are used to measure network throughput and delay, as well as monitor traffic. Chapter 4 details throughput performance relative to variable SNR, packet length, and window size. This chapter also describes the relationship between delay and packet length packet loss. The most significant contribution of

this chapter is the development of empirical models derived from experiment results. Chapter 5 provides a comprehensive conclusion.

Chapter2 Background

2.1 Introduction of 802.11ax

When deciding how to improve next generation wireless protocol, IEEE and Wi-Fi Alliance used a survey to ask users about Wi-Fi deployment and performance. Results showed that as many access points get deployed in a given location (e.g., university campus, apartment buildings, busy airports and train stations), congestion and interference are the leading cause for low data rate. Such sites have multiple access points and overlapping areas. To mitigate this problem, IEEE and Wi-Fi Alliance suggest that the new wireless protocol should focus on real-world scenarios rather than enlarging bandwidth, higher order MCS, or MU-MIMO. When compared with 802.11ac, 802.11ax promots the following novel features: operates on 2.4GHz band in addition to the 5.0GHz ; furnishes new ODFM symbol, provides downlink and uplink OFDMA, offers uplink MU-MINO (up to eight clients), is characterized by 1024 Quadrature Amplitude Modulation (QAM), and saves client battery life and spatial reuse (i.e., Basic Service Set Coloring [BSS color]).

OFDMA

OFDMA is the main 802.11ax enhancement, which enables simultaneous small frame multiple data transmissions. OFDMA divides a channel into multiple subcarriers. For example, in 802.11ac, 64 subcarriers of 20MHz bandwidth, with each subcarrier spacing 312.5KHz. One symbol takes at most 3.2usec with 0.4 or 0.8usec for Guard Interval (GI). 802.11ax is characterized by 256 subcarriers, with

each subcarrier spacing 78.125KHz so that one symbol duration increases to 12.8usec with 0.8 GI. These subcarriers are combined as sub-channels and are typically referred to as Resource Units (RU). When multiple transmissions utilize a 20MHz channel, 802.11ax access point (AP) allocates 26, 52, 106 and 242 subcarriers to the senders, which roughly equals to 2MHz, 4MHz, 8MHz and 20MHz channels, respectively. The AP allocates how many RUs are used, as well as the combination pattern for both uplink and downlink.

RU allocation information is maintained in the PHY layer and MAC layer. At the PHY layer, this includes how the channel is partitioned into RUs and which users are assigned to each sub-channel. At the MAC layer, RU allocation information triggers the 802.11 frame and specifies each RU by a 7-bits flag. When 802.11ax AP must send packets to multiple 802.11ax clients, the protocol first contends for the medium, and then is awarded the transmit opportunity. Next, AP sends a multi-user request-to-send (MU-RTS) frame, which reserves the medium and synchronizes with clients. Clients will return a clear-to-send (CTS) in parallel with assigned RUs. After CTS are sent to AP, the latter will transmit assigned RUs to corresponding users' ID and follow up with a data packet. Clients will receive packets via their assigned RU and send Acknowledge (ACK) to the AP.

Uplink-OFDMA (UL-OFDMA) is similar to DL-OFDMA, although it is more complex and may require the use of as many as three trigger frames. Each frame is charged with obtaining information to synchronize time/frequency for multi-users. Like DL-OFDMA, the AP must win a transmit opportunity and send the first trigger frame, buffer status report poll (BSRP), asking clients the number of buffers in transmit

window. Clients will respond with buffer status reports (BSR) enabling AP to leverage this information and best allocate RU. The second trigger frame, namely MU-RTS, allocates RU to each client. The third trigger frame signals clients to begin uplink transmission via unique RUs. Uplink window size and power control information is also included so that each client is able to increase or decrease transmit power and help AP reception rate.

MU-MIMO

MU-MIMO is another important 802.11ax enhancement. Both OFDMA and MU-MIMO support simultaneous multi-user transmission in both downlink and uplink. Notably, OFDMA is used for transmitting small frames, MU-MIMO is efficient in transmitting large packets. MIMO is also referred to as “spatial steams,” as it is based on the characteristic of radio propagation: multipath. MU-MIMO only operates when a client or group of clients do not hear a significant signal strength from other clients, and vice versa. In this way the AP can simultaneously transmit data frames to multiple client groups.

Downlink MU-MIMO transmission utilizes a NULL packet for acquiring knowledge of receiver signal strength for AP and all clients. AP then calculates a matrix that is applied to transmit power for decreasing loss rate at the receivers. 802.11ax downlink MU-MIMO is derived from 802.11ac and increases four spatial steams to eight. However, uplink MU-MIMO is a new 802.11ax feature in which downlink OFDMA is a more complicated process.

As the AP must discover client transmission information, the downlink MU-MIMO requires multiple trigger frames to simultaneously coordinate multiple client transmit. Trigger frames offer clients' MIMO groups and MCS value, as well as optimal transmit power strength. AP will then send a basic trigger frame informing when and how clients should response.

1024-QAM

Also on the physical layer, 802.11ax uses higher modulation rate for 1024-QAM in favor of 256-QAM in 802.11ac. 256-QAM carries eight bits in one symbol, whereas 1024-QAM carries 10 bits. Data rate will increase by 25% in the high receive signal strength index (RSSI) condition, primarily because the decision space between adjacent points on 1024-QAM constellation decreases 25% more than in 256-QAM.

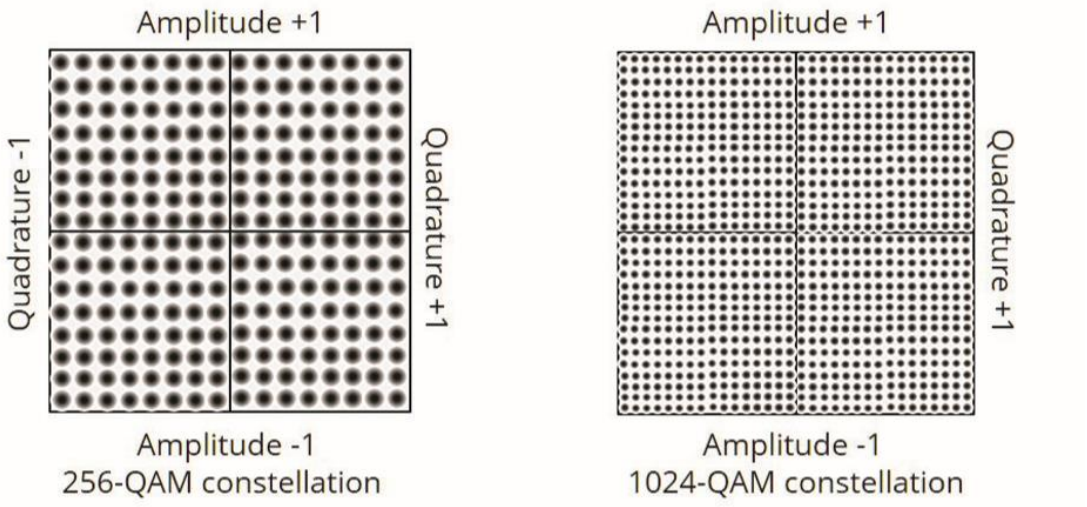

Figure 2.1. Constellation diagram for 256 and 1024 QAM. Adapted from "White Paper 802.11ax', n.d. retrieved from https://www.arubanetworks.com/assets/wp/WP_802.11AX.pdf

Target Wakeup Time (TWT)

IoT has grown into a big market with most devices embedded with a Wi-Fi modules. Although LTE is used for handling date generated by Wireless Sensor Network (WSN), it is unable to manage a large date exchange. WLAN is promoted to become the prominent wireless standard for IoT. 802.11ax implements target wakeup time (TWT) to save battery life of sensors. It also can serve multiple sensors in low date rate with short packets. TWT allows client and access point to negotiate a schedule for client wakeup and communication. When scheduled time arrives, client awakens and waits for the polling request to be sent from access point along with transmits date. After transmission, client returns to idle/sleep state.

BSS Coloring

Wi-Fi is characterized by a medium access control protocol, namely CSMA/CA, to control transmit opportunity for uncoordinated users. Users first sense the air when preparing to transmit a packet. Given that a user senses energy above a certain power threshold, user defers transmission and leverages the back-off window. CSMA/CA permits Wi-Fi devices to communicate with one another without previous coordination. This saves a significant amount of time and energy. However, sites with highly dense AP's, nearby AP's will co-channel interfere with each other under heavy channel usage, making CSMA/CA to have very low capacity. BSS coloring was introduced in 802.11ax. This technology operates by

differencing “same color” user and “distance color” user, and then applying a different power threshold.

Chapter 3 Experiment Setup and Results

3.1 Hardware and software configuration

Experiments were designed to perform this study and conducted on the fourth floor of Building Four at the Oklahoma University-Tulsa campus. An 802.11ax network was set by the router, which was connected to Mac via a 1Gbps Ethernet cable and a 1Gbps Ethernet adapter. A Lenovo laptop was equipped with an Intel ax200 wireless card that supported 802.11ax. The testbed is shown in Figure 3.1. Both laptops are not connected to the Internet to prevent any background Internet traffic and, limiting any interference with the throughput testing.

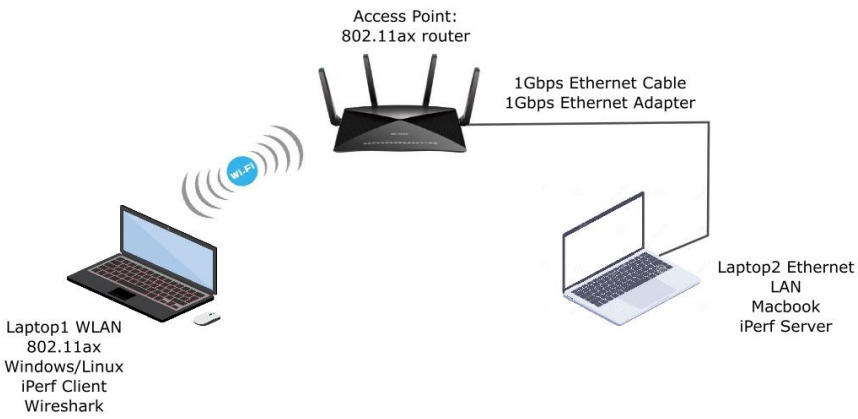

Figure 3.1. Experiment hardware setup.

Due to the fact that the router does not provide transmit power adjustment service, SNR varies by changing the distance between router and laptop. Tests were performed on weekends or in the evening to avoid interference with other wireless devices. Hence, background noise can be assumed minimum and constant during

testing. Network power measurement and performance testing were performed separately but at the same locations to avoid the negative influence or throughput biasing when both measurements are carried out at the same time. Given that tests are relocated to a new power measurement testing site, researches waited one or two minutes while the wireless card adjusts to an appropriate Modulation Coding Scheme (MCS) value. Researchers ensured that the test space was empty of other users, minimizing the effect of passing users on receive signal strength and throughput.

Network generation between laptop and router were produced by iperf3. Wireshark was used to monitor traffic status. Iperf3 is a tool designed to aid users in measuring maximum IP network bandwidth. The system simulates real-world TCP and UDP transmissions and analyzes network performance of throughput, jitter, and data loss rate. Netspot and Acrylic Wi-Fi were used to measure RSSI, and ASUS router measured noise.

The network diagnostic tool mtr was used to measure delay. Rather than provide simple round travel time (RTT) between the server and the client, mtr collects additional information about the channel state, connection state, and responsiveness of the intermediate hosts. Table 3.1 shows laptop configurations.

Table 3.1 Configuration of Laptops

	Lenovo ThinkPad	Lenovo ThinkPad	MacBook Pro
Operation System	Windows 10	Ubuntu 18.04.3 LTS	macOS Mojave
Iperf Version	3.1.3	3.7	3.7
SNA Tool	Acrylic Wi-Fi	N/A	Network Monitor
Network Interface	Killer ax1650	Killer ax1650	Ethernet Cable
Delay Tool	N/A	mtr	N/A

Chapter 4 IEEE 802.11ax Network Performance without Interface

This section presents 802.11ax network performance and empirical models developed from the experiment. TCP/UDP throughput measurements were acquired by varying SNR and investigating possible relationships. Matlab was used to visualize test data and minimum mean square error (MMSE) was used to derive the empirical model.

4.1 802.11ax Throughput Performance

Throughput measurements were determined using TCP/UDP transport protocol generated by iperf3 with different SNR. Parameters are set as default. Results are shown in Figure 4.1.

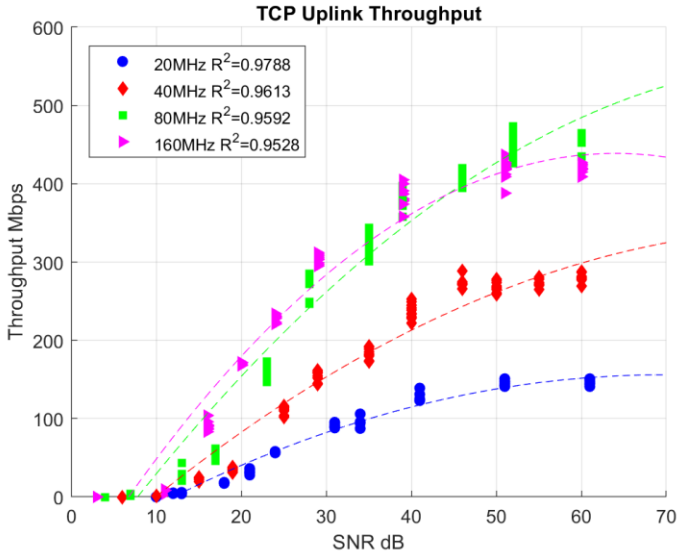

Figure 4.1. 802.11ax uplink throughput, TCP.

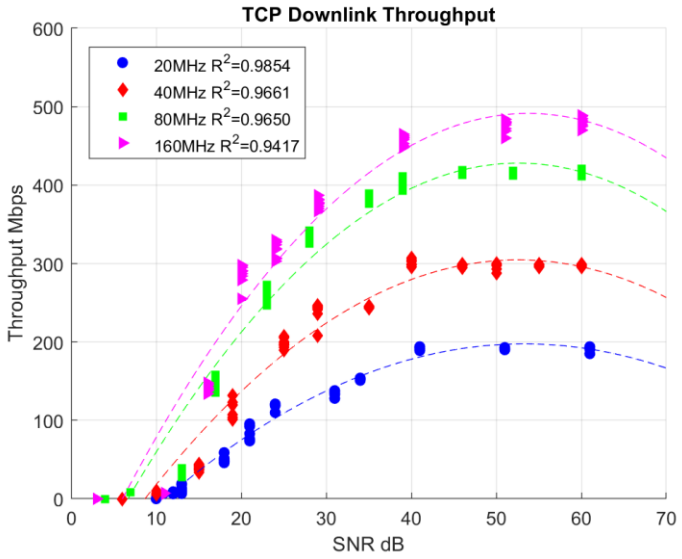

Figure 4.2 802.11ax. Downlink throughput, TCP.

MATLAB was used to examine data calculated by iperf3. Following is an 802.11ax throughput second order polynomial model:

$$f(x) = p_1 \times x^2 + p_2 \times x + p_3$$

where x is SNR, and p1, p2, and p3 are coefficients calculated from MATLAB's fit command (See Table 4.1 and 4.2).

Table 4.1 Parameter of 802.11ax Uplink Throughput

	p_1	p_2	p_3
20MHZ	-0.04714	6.555	-72
40MHZ	-0.00557	9.86	-92.47
80MHZ	-0.08372	14.96	-111.9
160MHZ	-0.1338	17.14	-109.9

Table 4.2 Parameters of 802.11ax Downlink Throughput

	p_1	p_2	p_3
20MHZ	-0.1107	11.79	-116.6
40MHZ	-0.158	16.62	-132.4
80MHZ	-0.2032	21.37	-133.9
160MHZ	-0.2157	23.2	-109.9

Figures 4.2 and 4.3 demonstrate that throughput increases as the bandwidth and SNR increase, with the exception of the 160MHz uplink. This uplink's throughput has behavior similar to 80MHz but with maximum throughput lower than 80MHz. To determine why 160MHz did increase throughput, data for 160 and 80MHz were tested separately and measured over a longer period of time. Figure 4.2 shows downlink throughput and Figure 4.3 shows uplink throughput when bandwidth was 160MHz (blue line) or 80MHz (orange line).

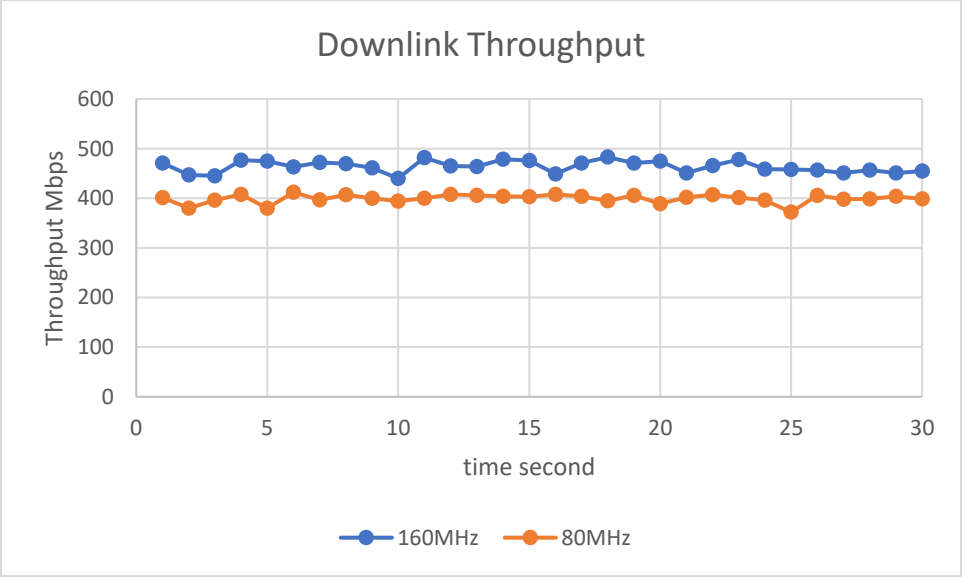

Figure 4.3. 80MHz and 160MHz downlink throughput.

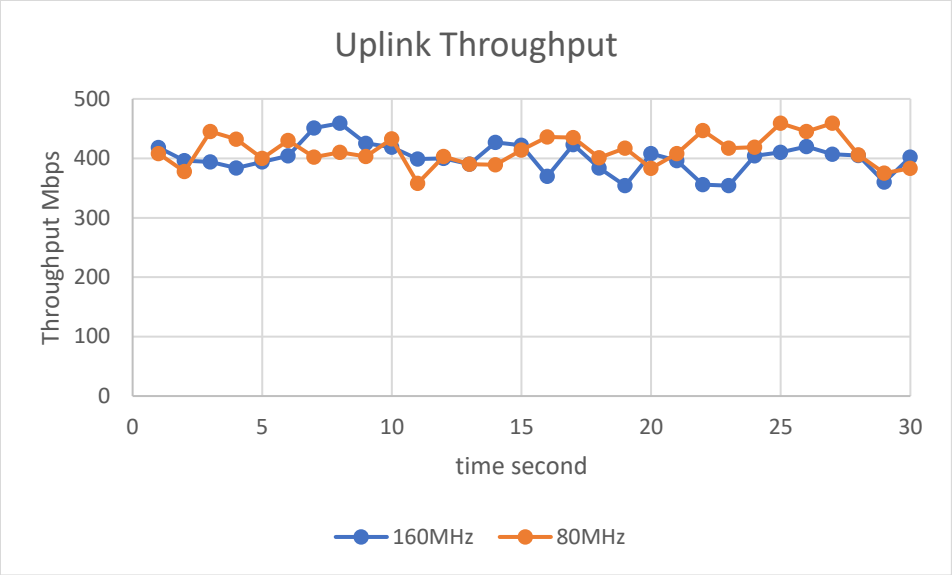

Figure 4.4. 80MHz and 160MHz downlink throughput.

In this testing scenario, RSSI was equal to -30dBm. The figures above indicate that uplink throughput does not benefit from doubling the bandwidth.

UDP's maximum throughput is greater than that of TCP because UDP is not a reliable, error-correction protocol and not limited to a windowing used in TCP. When starting a TCP transmission, client will build a connection with a server by three-way handshaking (i.e., "SYN, SYN-ACK, ACK"), as three messages are required. This process is designed to initiate and negotiate with initial sequence number (ISN) at both ends to establish reliable, bi-directional communication. During the transmission period, client and server will track received packets by using sequence number, and receiver will respond with ACK, which tells sender the received packet's sequence number. Given that sender does not receive ACK after timeout or receives a duplicated ACK due to packet loss, TCP will request retransmission. When TCP detects congestion in the medium, the system will start

congestion control, decrease sender socket, and cause a decrease in throughput. UDP is an alternative communication protocol used primary for establishing low-latency and loss-tolerating communication between server and clients. It does implement retransmission, error correction, nor flow control mechanism; thus it is characterized with a much lower bandwidth overhead, as well as latency. UDP works by encapsulating data in UDP packet and adding a header to the packet. Header information contains source and destination port number to facilitate communicate, packet length, and checksum for error checking. After UDP packets are encapsulated in an IP (internet protocol) packet, the information is transmitted to the destination. Since UDP is an unreliable, fast transmission protocol, the packet is always utilized in an application requiring a large amount of data and small latency (i.e., videos or online games).

Figure 4.6 indicates that saturation UDP throughput of 802.11ax is above 700Mbps, which is 40% more than the TCP downlink. Throughput saturation occurs when SNR is larger than 60dB. Reduction in throughput becomes due to power saturation at the RF-front end. A second polynomial model was calculated to describe 802.11ax UDP throughput behavior. Table 4.3 shows coefficients of second order polynomial.

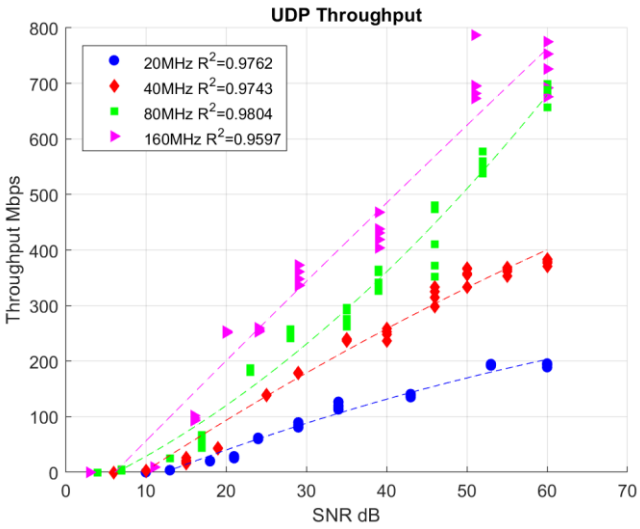

Figure 4.5. 802.11ax throughput, UDP.

Table 4.3 Parameters of 802.11ax UDP Throughput

	p_1	p_2	p_3
20MHZ	-0.02473	6.054	-71.05
40MHZ	-0.02827	9.938	-93.98
80MHZ	0.09671	6.237	-43.33
160MHZ	-0.00823	14.7	-90.05

Throughput vs Packet Size

MTU (maximum transmission unit) is identified as the largest chunk of packet transmitted by transport protocol across the air. Default value is 1500 bytes for most network interface and routers. Any packets larger than 1500 bytes must be fragmented into smaller pieces for transmission. TCP MSS (Maximum Segment Size) is equal to MTU minus 40 bytes (i.e., 20 bytes for ipv4 header and 20 bytes for TCP header), so that default MSS is 1460 bytes. Standard MTU remained fixed for backward compatibility with the previous network. Because the header is constant, it is obvious that small packet size has a lower throughput. SNR is equal

or larger than 60dB during the test to ensure that noise has no influence on throughput performance. The relationship between packet size and throughput is shown in Figure 4.7.

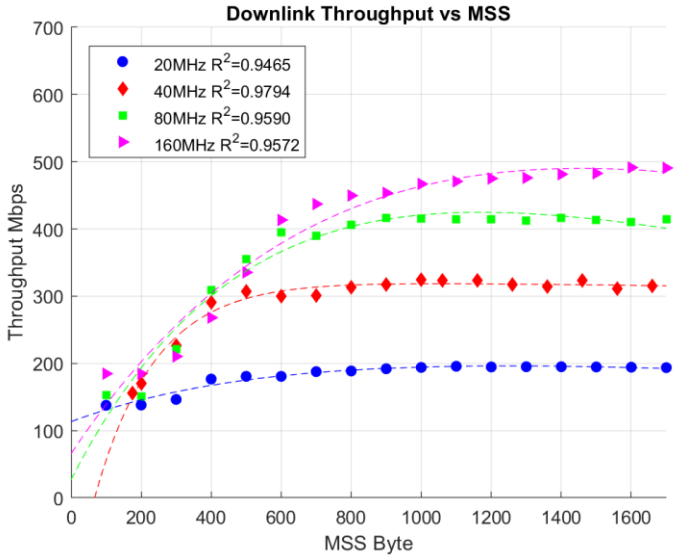

Figure 4.6. 802.11ax throughput with different segment length.

MSS was modified on the network interface and monitored by Wireshark to ensure all data packet payloads were equal to or less than 1460 byte. Because Wireshark was installed on the client laptop, transmission must be initiated by the server. Results are matched with the downlink throughput shown above. Saturation throughput is 200Mbps, 300Mbps, 410Mbps, and 500Mbps at 20MHz, 40MHz, 80MHz, and 160MHz, respectively. Critical packet size to approach saturation is 1200 bytes at 160MHz and 800 bytes for others bandwidth. Figure 4.8 shows packet length and throughput in 15 second increments. The figure also indicates that throughput gradually increases from zero to one second. After this threshold, transmission rate increased until saturation approach, due to TCP mechanism. Given that TCP transmission is generated after the three-step hand shake, sender

does not have knowledge about receiver's buffer size. Hence, the initial congestion window size is one MSS. Given that sender receives a successful Ack response and packet loss did not occur, sender doubles the congestion window and grows exponentially. This process is referred to as "slow start," although it not slow. The increment in size is continuous until sender detects packet loss or congestion window searches a threshold (i.e., slow-start threshold [ssthresh]). The algorithm then enters a new phase, namely congestion avoidance. Given that sender's timeout doesn't expire or duplicative Ack are not received, the congestion window was increased by one MSS for each round-trip. After two seconds, throughput change levelled off.

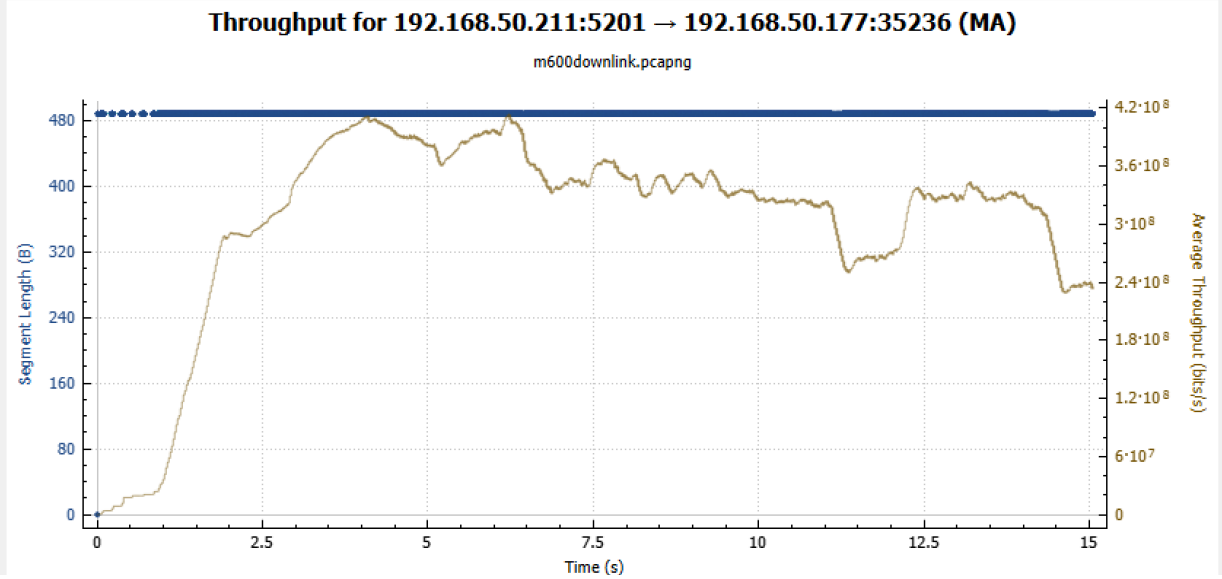

Figure 4.7. Segment length during TCP transmission and real time throughput.

MSS was set to 500 bytes. Since the timestamp was enabled during connection, the TCP header increased to 32 bytes. Thus, segment length was 488 bytes; 12 bytes indicates the time when packets were transmitted from the sender.

When bandwidth was 20MHz, throughput reached saturation 200Mbps at 800 bytes. When bandwidth was 160MHz, saturation 500Mbps occurs given that MSS is 1400 bytes (See Figure 4.8). This phenomenon indicates that a higher capacity network optimizes its performance with larger packet size. However, the result is an increase in packet size with no fragmentation between server and client. Throughput will decrease because the required transmission environment must be suitable (e.g., RSSI is higher than a threshold; versus, a lossy environment). High error rate and loss rate requires server to retransmit lost packets, which decreases throughput. Another factor is the limitation in receiver socket size. During communication in TCP between server and client, they exchange window buffer size to calculate the number of bytes in the subsequent data exchange. Receiver socket size will be filled with a jumbo packet given that its read speed is not fast enough to process a sufficient number of packets. Server will then receive an ACK indicating “the receiver socket window is full, stop transmission.” Bandwidth is absent during wait time, and overall throughput decreases.

An exponential model was calculated to describe the relationship between throughput and MSS. The equation is as follows:

$$f(x) = ae^{bx} + ce^{dx},$$

where $f(x)$ is the throughput, and x is referred to MSS. Table 4.4 shows the coefficient of the exponential model.

Table 4.4 Parameter of 802.11ax Throughput vs MSS

	a	b	c	d
20MHZ	262.2	-0.0001519	-149.3	-0.001565
40MHZ	326.6	-2.2053e-5	-480.4	-0.005757
80MHZ	777.6	-0.0003309	-750.6	-0.001695
160MHZ	161400	-0.0006503	-161400	0.0006508

Throughput vs Window Size

Each network socket is allocated a send buffer for outbound packets and a receive buffer for inbound packets. Buffers are assigned a default size determined by operating system parameters. One drawback in setting a static value to the TCP buffer is that network performance is inherently dynamic. A small buffer size will underutilize the network, and a large buffer size will waste RAM while blocking other applications. Thus, Windows- and Linux-imposed TCP Buffer Auto Tuning, sometimes is referred to as Dynamic Right-Sizing, is often utilized to adjust TCP buffer size.

Because Windows is not supported to change socket size, a virtual Linux system was installed for testing. Linux has two parameters for controlling the value of send and receive socket size: *net.core.wmem.default* and *net.ipv4.tcp.wmem*. The *net.ipv4.tcp.wmem* is 4096 131072 6291456, indicating that 4096 is the minimum send buffer size for a single TCP socket; 131072 is the default TCP send buffer size; and 6291456 is the maximum TCP send buffer size. The receive buffer size

is identical to the send buffer. By looking into the iperf3 debug mode, the minimum send socket size is 2048 bytes, which is half of the minimum TCP send buffer size due to the TCP mechanism. Minimum receive socket size is 4096 bytes. Notably, maximum buffer size is actually bounded by the *net.core.wmem.default and net.core.rmem.default* (i.e., default receive buffer size). On the Linux operating system, the control command is used to set socket size, which was 32MB in the test reported herein. Figure 4.9 and 4.10 show throughput behavior as a result of changing socket size. Figure 4.9 illustrates the result when send buffer size is changed; Figure 4.10 shows results for receive buffer size. RSSI remained above -30dBm during the entire test; thus, throughput achieved maximum value.

Figure 4.9 illustrates results due to a change in send socket size.

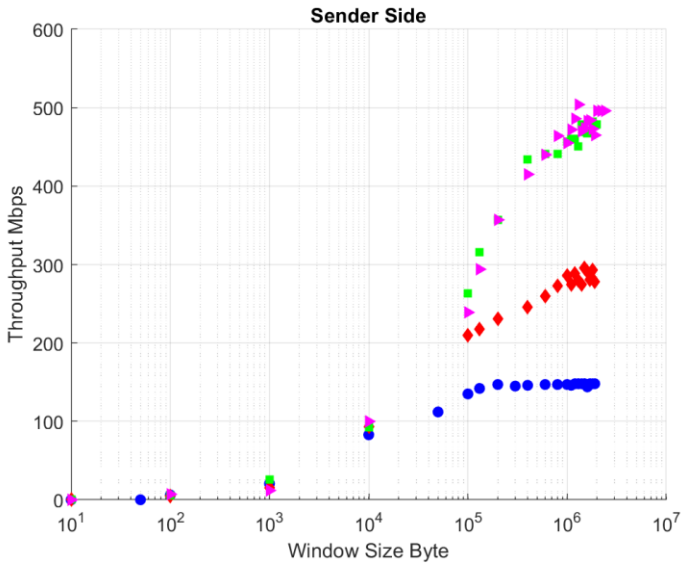

Figure 4.8. 802.11ax throughput with various sender buffer sizes.

Figure 4.10 illustrates changes in receive socket when server transmitted data to client.

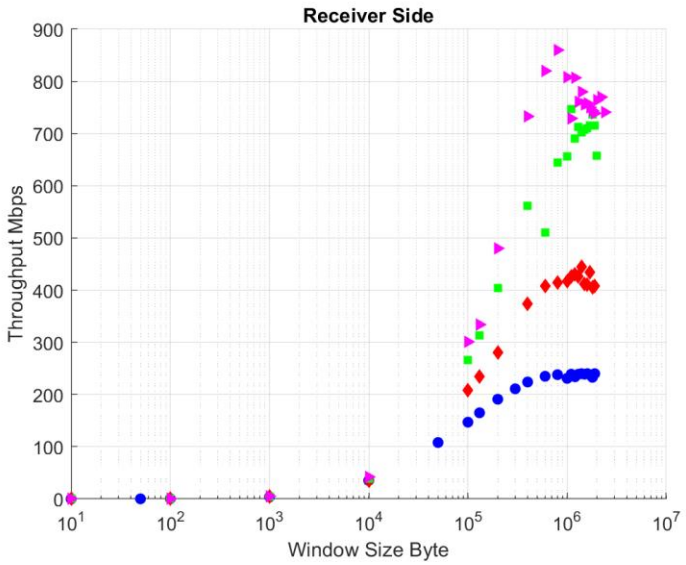

Figure 4.9. 802.11ax throughput with various receiver buffer sizes.

For an application to transmit data to a distant computer using TCP, it must first use send()/writer() function to copy data into the send socket. Next, TCP must transmit data to the destination computer and save the information in the receive socket. The send buffer retains data until an ACK response is detected by the receiver. In short, send socket size (i.e., window size) controls the number of bytes transmitted before sender receives an ACK. Figure 4.11 shows throughput performance for 802.11ax when send window size changes and receive window size is default value. This figure clearly demonstrates that as the send window size increases, throughput also increases until the point of saturation.

Receive buffer will save data following transmission until the read()/recv() function is utilized to copy data to the application. Buffer will accumulate data if read()/recv() is not activated. TCP flow control requires that a mechanism is in place to prevent a sender from overwhelming a receiver. Response ACK contains information about

remaining space in the receive buffer. Given that the receive buffer is full, the server will cease sending data to the socket until receiver window size is greater than zero. Figure 4.11 illustrates that throughput performance in four bandwidths remains the same until receive buffer size is larger than 10000 bytes, at which time throughput increases to saturation state.

Notably, Windows Buffer Auto-Tuning is not fully utilized in the channel. Maximum downlink data rate achieved 880Mbps when the receive buffer was fixed to 1000,000 bytes in Linux and 500Mbps in Windows.

Windows interface utilizes CUBIC to adjust congestion window size at the congestion avoidance phase. The CUBIC [7] algorithm developed from the BIC (Binary Increase Congestion) algorithm is less aggressive and more convenient for replacing the concave and convex window update. A key feature of CUBIC is window growth independent of RTT. This feature allows fair transmission on the same bandwidth, even if each has a different RTT. Figure 4.11 shows window growth as a result of the BIC and CUBIC algorithm. Packet loss indicates that data traffic on the internet experiences congestion at various points. To avoid packet loss, sender will reduce its congestion window size and search for a stable window size without packet loss.

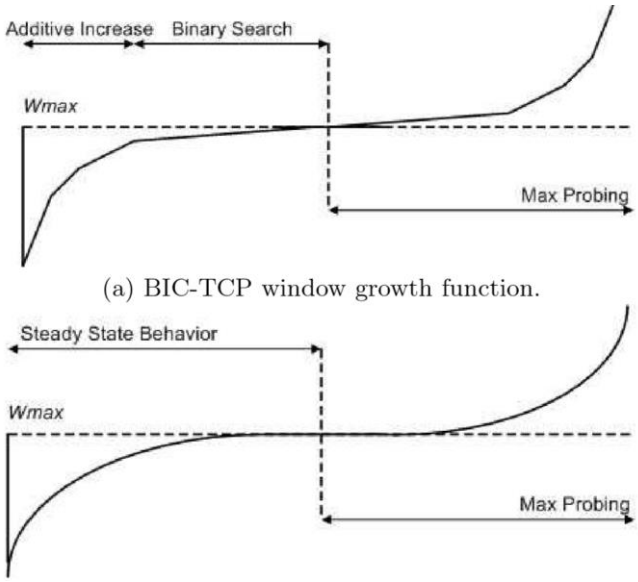

Figure 4.10. CUBIC window growth function Adapted from" CUBIC: A New TCP-Friendly High-Speed TCP Variant", Sangtae, Injong, Lisong, 2008.

Given that a packet loss event is detected by the sender, BIC will reduce its windows size by multiplicative factor β. Window size prior to packet loss is set to W_{max}, and window size immediately following reduction is set to W_{min}. BIC algorithm will update the congestion window by advancing to the midpoint between W_{max} and W_{min}. Given that window size grows larger than the maximum size, a new maximum window size must be searched (i.e., "max probing" phase). The max probing window growth function first increases slowly when finding the new maximum. After some time, however, and given that no packet loss occurs, the max probing window can safely assume that the new maximum is further away. In this way, BIC switches to a fast increment phase by increasing window size using a larger fixed value. Although BIC achieves good stability in high speed network, the algorithm is unfair to the traffic with small RTT and a high computation cost. A

stream with a shorter RTT-updates congestion window is quicker than a stream with longer RTT. Hence, the longer RTT stream occupies only a small part of resource block, decreasing user's QoS (Quality of service).

Linux will implement the CUBIC congestion control algorithm after 2.6.19. Cubic sets the window size just before packet loss as inflection of the cubic function. Like BIC, the send window size first decreases with a multiplicative factor β. After it enters the congestion avoidance phase from fast recovery, the CUBIC window size grows like a concave curve until it approaches W_{max}. After that, window size growth is expressed like a convex curve. This style of window adjustment (i.e., concave and then convex) improves protocol and network stability while maintaining high network utilization. The window size growth function of CUBIC has the following term:

$$\mathrm{W}(T) = \mathrm{C}(t-K)^3 + W_{max},$$

where C is a CUBIC parameter; t is the time following the last window reduction; and K is the time period in which the window size increases from W to W_{max}, given no packet loss. K is calculated using the following equation:

$$\mathrm{K} = \sqrt[3]{\frac{W_{max}\beta}{C}}$$

The equation above, along with the symmetric characteristic of CUBIC function, illustrates that C controls the speed at which the congestion window grows to W_{max} and that β controls the scale.

Jitter vs SNR

Jitter can be conceptualized as the difference in received packet delays. At the sending side, packets are transmitted in a continuous stream with evenly distributed space between each packet. Network congestion or queuing delay in routers causes the space between each packet to vary. If jitter is too high, in voice or video stream the jitter buffer is made large to compensate. Figure 4.12 shows the relationship between receiver's SNR and jitter measured using UDP. It also shows the standard deviation of each measurement.

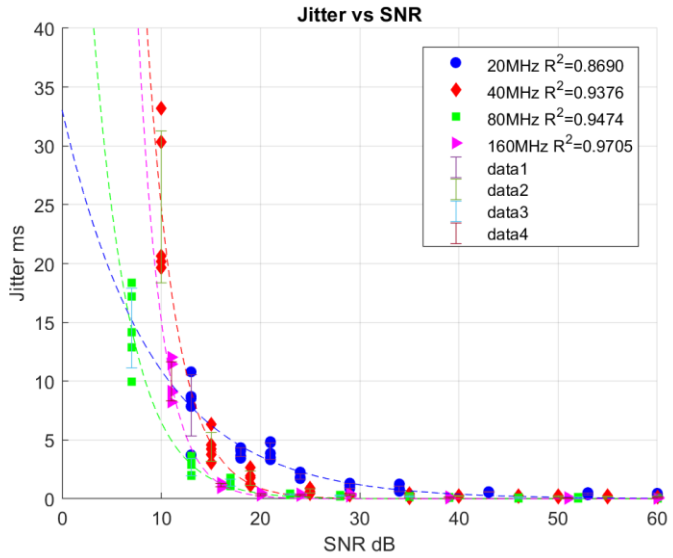

Figure 4.11. 802.11ax jitter with various SNR.

Overall, jitter in four bandwidths decreased as SNR increased. Jitter can be described by two phenomena. First, jitter remains lower than 5ms when SNR increases from 20dB to 60dB—although jitter is typically shorter than 1ms. Second, the jitter curve can be characterized by an exponential behavior, where the highest value is 33ms.

Based on the testing setup, the router served only two laptops. Therefore, one can surmise that jitter is mainly due to packets that are lost as a result of low SNR.

An exponential model was derived to describe the relationship between Jitter and SNR. The equation is, as follows:

$$f(SNR) = a \times e^{(b \times SNR)}$$

Table 4.5 details the coefficient of the exponential model.

Table 4.5 Parameter of 802.11ax Jitter vs SNR

	a	**b**
20MHZ	33.03	-0.1108
40MHZ	685	-0.3319
80MHZ	93.35	-0.2663
160MHZ	991.5	-0.4181

4.2 802.11ax Delay Performance

Delay is yet another critical problem for wireless network throughput, especially in today's wireless market. Latency sensitive applications, like Skype, FaceTime, or even live TV, attempt to provide low-latency connection for end users. Delays result not only from radio propagation, but also from retransmission, medium contention router processing, and local buffering. This concept can be understood by the following statement:

$$D_{e2e} = D_{propagation} + D_{processing} + D_{transmission} + D_{queuing}$$

$D_{propagation}$ is the length of time required for bits to propagate through a specific medium from sender to receiver. For wireless communication, propagation velocity is the speed of light c. $D_{propagation}$ can be calculated using the following equation:

$$D_{prpagation} = \frac{d}{c},$$

where d is the distance between two end users.

$D_{processing}$ is the time required by intermediate routers to decide where to forward a packet, update time to live (TTL), and calculate checksum. This factor is often neglected because the value is much smaller than other components.

$D_{transmission}$ is the amount of time required to push all bits on the transmission medium. Transmission can be calculated using the following equation:

$$D_{transmission} = \frac{N}{R}$$

where N is number of bits, and R is transmission rate in bits per second.

$D_{queuing}$ is the time a packet waits in a queue before it can be executed. Notably, a router can process only one packet at a time. If packets arrive faster than the router can process, the router must place packets into a queue (i.e., buffer), until that time at which the router can transmit them. If the queue is full, the router will throw away excess packets, which causes packet loss.

Retransmission also causes delay between sender and receiver. Because TCP is a reliable communication protocol, it has two methods for activating retransmission: Retransmission timeout (RTO) and Duplicate cumulative acknowledgements (DupAcks).

4.2.1 Retransmission timeout

When the sender transmits a packet, a retransmission timer starts, and then resets given that the ACK is received. If the timer expires the retransmission timeout, the sender retransmits the packet and decreases congestion windows. The retransmission timer then doubles its value. The detail calculation process is in RFC6298. Linux will modify parameters to better fit the network.

Two parameters are used to compute current RTO: a) roundtrip time variation (RTTVAR) and b) smoothed roundtrip time (SRTT)

Until an RTT measurement has been achieved between sender and receiver using the timestamp, the sender must set initial RTO as 1 second:

$$RTO = 1$$

When the first RTT measurement R is complete, the sender establishes

$$SRTT = R$$

$$RTTVAR = \frac{R}{2}$$

$$RTO = max(RTO_MIN, 4 \times RTTVAR)$$

The minimum RTO is Linux in 200ms, and the maximum RTO is 120s.

When subsequent RTT measurement " R' " is made, the sender set is characterized by

$$RTTVAR = (1 - \beta)RTTVAT + \beta \mid SRTT - R' \mid$$

$$SRTT = (1 - \alpha)SRTT + \alpha R'$$

In RFC6298, $\alpha = \frac{1}{8}, \text{and } \beta = \frac{1}{4}$. Linux modified these two coefficients when $\mid R' - SRTT \mid > RTTVAR$, meaning R' is fluctuating to quickly, so that β is set to $\frac{1}{32}$ to smooth RTO.

If $x_{n-1} = SRTT_{n-1} - R_n$, then RTTVAR and SRTT can be written as

$$RTTVAR_n = \left(\frac{3}{4}\right)^n RTTVAR_0 + (\frac{3}{4})^{n-1} \times \frac{1}{4} \mid x_0 \mid + (\frac{3}{4})^{n-2} \times \frac{1}{4} \mid x_1 \mid + \ldots + \frac{1}{4} \mid x_{n-1} \mid$$

$$SRTT_n = \left(\frac{7}{8}\right)^n SRTT_0 + \left(\frac{7}{8}\right)^{(n-1)} \times \frac{1}{8} R_1 + \left(\frac{7}{8}\right)^{(n-2)} \times \frac{1}{8} R_2 + \ldots + \frac{1}{8} R_{n-1}$$

The above equation reasons that RTTVAR is the weighted sum of variation between current RRT and previous SRRT. Therefore the RTTVAR distribution estimate is based only on the algorithm, because it depends on the order of RTT.

4.2.2 Duplicate cumulative acknowledgements

An alternative way to retransmission loss packet is DupAcks. TCP uses a sequence number to identify each data frame. Sequence number identifies the order data so that the receiver can reconstruct date in order. Given that the receive side receives a packet, it will respond with an Ack contained the next sequence number of the packet. If packet loss was inadvertent, the sender will receive a duplicate response and understand a packet loss occurred during transmission. A lost pack message will be sent (See Figure 4.13).

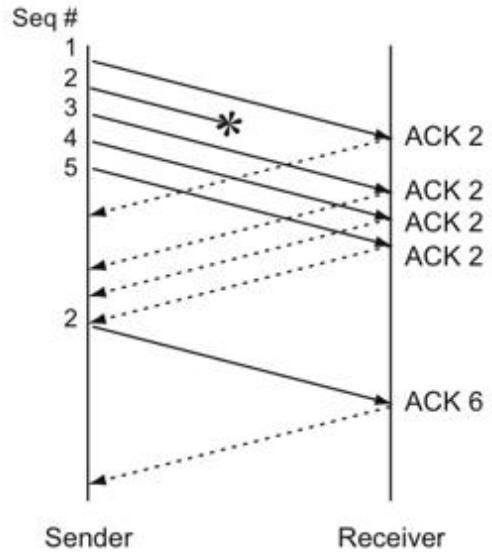

Figure 4.12. Duplicate acknowledgement and retransmission Adapted from "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems", Phanishayee et al, 2008.

Figure 4.13 demonstrates that the sender transmitted packet no.1 and responded with Ack2. However, because packet no.2 was lost, the receiver didn't receive the frame and repeated response Ack2. Sender transmitted packet no.3, no.4, and no.5, and repeatedly received Ack2. Sender understood packet no.2 was lost and retransmitted. However, sender didn't know the receive status of packet no. 3, no.4. and no.5. Hence, it will retransmit these packets and degrade throughput. In order to only retransmit the lost packet, TCP uses a new function called selective acknowledgements (SACK). If both server and client support SACK, TCP header will also contain SACK information, indicating the boundary of received packets. Server will only retransmit lost packets. Unlike RTO, the congestion window does not decrease to 1 given that DupAcks occurs, as it is a less aggressive mechanism. This process is referred to as fast retransmission.

Figures 4.14 and 4.15 show CDF (cumulative distribution function) of RTT (round travel time) with different packet sizes. Packet loss was zero during the test. Packet size was set to 700 bytes and 1400 bytes; transmission rate varies from 10 packets per second to 20000 packets per second.

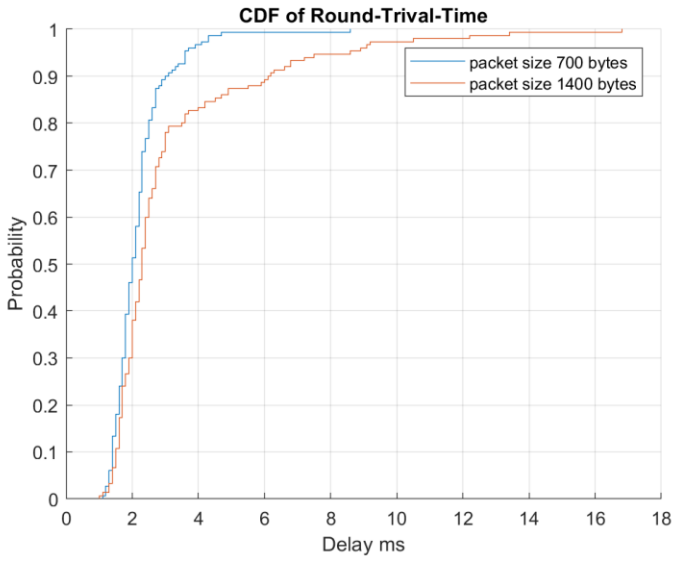

Figure 4.13. 700 bytes and 1400 bytes packet cumulative distribution of RTT.

Delay was measured using mtr, which can test traffic between client and server. Unlike ping, which only sends network Internet Control Message Protocol (ICMP) throughout the network, mtr uses network TCP and UDP through the network to test RTT. Mtr provides router delay statistical information available on client and server.

The figures above show RTT distribution with various packet size. A 700 bytes packet had the lowest average RTT, while the 1400 byte packet had the highest RTT. Transmission delay is proportional to frame length. The computer requires more time to transmit a 1400 bytes frame to the medium.

Understanding that TCP is a reliable protocol, one can be confident that if packet loss is greater than zero, average end-to-end transmission time increases. Throughput decreases to zero when packet loss is 100%.

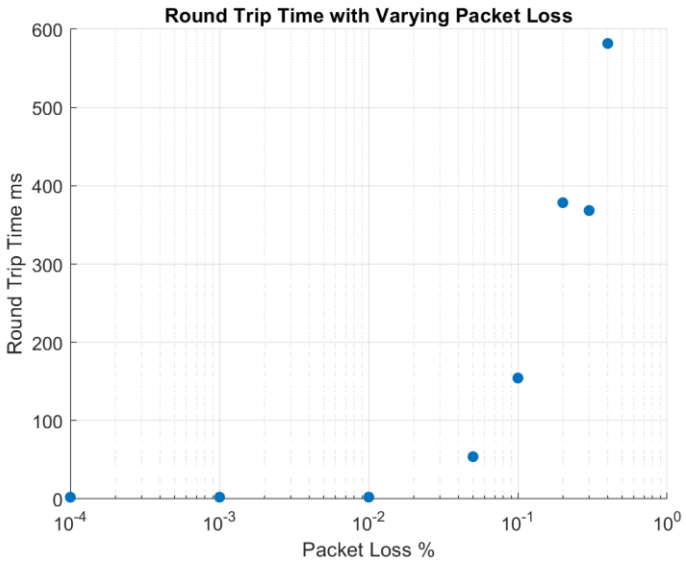

Figure 4.14. Round-Trip time with different packet loss.

Given that packet loss is less than 1%, round-trip time is similar to zero packet loss. After 1%, round-trip time will increase.

4.3 Conclusion

The work reported in this chapter tested and discussed IEEE 802.11ax network performance and factors that influence performance. Empirical models were built to evaluate the way in which factors affect IEEE 802.11ax performance. Results showed that receiver signal strength-to-noise ratio, frame size, and sender/receiver window size are critical for determining throughput. According to Shannon's Theorem (i.e.,$C = Blog_2(1 + SNR)$), theoretical upper bound bit rate is controlled by SNR. Hence, to increase throughput for a given bandwidth, SNR should be increased. After careful measurement and adjustment, models for SNR, packet length, and window size were developed. When using models, researchers can approximate 802.11ax throughput. Figures and mathematical models demonstrate that network performance always has two inflection points—one

when throughput increases from zero and another when throughput enters the saturation phase. Also, delay performance was estimated in this chapter, indicating the lower the packet loss, the lower the RTT between server and client.

Chapter 5 Conclusion and Future Research

All tests reported in this thesis were conducted on the University of Oklahoma-Tulsa campus. Empirical data indicated that 802.11ax throughput can be modeled and predicted as a function of SNR. Empirical models were able to approximate the network throughput when SNR is tested. Results showed that throughput reaches saturation when SNR is larger than an inflection point. In this case, throughput behaves like a quadratic function of SNR. Additionally, estimated packet size and window size suggested that they also have impact on the throughput. As packet size or window size increases, throughput also increases and eventually reaches saturation state. Small packet throughput was shown to be less than large packet throughput. Smaller packets have relatively high overhead as compared to its payload, hence throughput at the application layer is degraded by small packet's overhead. Most network interfaces and routers set MTU equal to 1500 bytes, although some interfaces support transmitting a packet with 9000 bytes. However, given transmitting 9000 bytes in one packet, SNR must be high to decrease bit error rate. In a lossy environment, a long frame is vulnerable in erroneous channel. Receiver will discard a packet given that error rate is higher than receiver can tolerant.

The analysis indicates that as window size increases, throughput also increases until the point at which it reaches saturation. Window size is controlled by TCP to protect the network from becoming overwhelmed and severly congested.

Furthermore, this thesis evaluated and discussed the SNR factor relative to jitter. Jitter was shown to be exponentially related to SNR. This phenomenon is caused

by packet loss when SNR is low. In this case, sender must retransmit lost packets. so that the variation between two packets' timestamp exponentially increases. Finally, this thesis tested the effect of packet length and packet loss on network delay.

Based on measurements reported in this thesis, one can see that there are several possible areas for the future work. First, researchers could study RTT with respect to SNR and build an empirical model based on the resulting data. Second, additional measurements can be made to study and model the MIMO scheme on network throughput and delay and to evaluate the improvement of throughput as multiple antennas are added to the access point.

References

[1] https://www.arubanetworks.com/assets/wp/WP_802.11AX.pdf

[2] Qiao Qu, Mao Yang, Zhongjiang Yan, Bo L (2015)i, "*An OFDMA based concurrent multiuser MAC for upcoming IEEE 802.11ax"* , Northwestern Polytechnical University, Xi'an

[3] Jorden Lee (2018), "*OFDMA-based Hybrid Channel Access for IEEE 802.11ax WLAN"*, Limassol, Cyprus

[4] Arjun Malhotra, Mukulika Maity, Avik Dulta (2019), "*How much can we reuse? An empirical analysis of the performance benefits achieved by spatial-reuse of IEEE 802.11ax"*, Dep. of CSE, IIIT, Delhi, Bengaluru, India, India

[5] Zineb Machrouh, Abdellah Naijd (2018), "*High efficiency IEEE 802.11ax MU-MIMO and frame aggregation analysis*", Marrakech, Morocco

[6] M. Shahwaiz Afaqui ; Eduard Garcia-Villegas ; Elena Lopez-Aguilera (2017), " *IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi"*, COMSATS University, Pakistan

[7] Sangtae Ha, Injong Rhee, "*CUBIC: A New TCP-Friendly High-Speed TCP Variant"*, North Carolina State University Raleigh, NC