PERFORMANCE IMPROVEMENT STRATEGIES FOR 802.11 NETWORKS

A Thesis

by

Kıvanç Çakmak

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Electrical and Electronics Engineering

Özyeğin University
August 2017

PERFORMANCE IMPROVEMENT STRATEGIES FOR 802.11 NETWORKS

Approved by:

Assoc. Prof. Ali Özer Ercan, Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Assoc. Prof. Fatih Uğurdağ
Department of Computer Science
Özyeğin University

Assoc. Prof. Onur Kaya
Department of Electrical and Electronics Engineering
Işık University

Date Approved: 18 August 2017

to Şengül Çakmak and Atife Arslanlar

ABSTRACT

802.11 networks have enabled and are fueled by the proliferation of connected mobile devices with a wide variety of applications, which have been integral parts of our daily lives. Currently, IEEE 802.11ac is a state-of-art protocol that provides higher data rates than its predecessors to accommodate the increasing demands of users with more bandwidth, spatial streams, modulation schemes, and downlink multi-user MIMO (MU-MIMO). In addition, 802.11ac protocol inherits aggregation and QoS prioritization, from 802.11n and 802.11e, respectively. On the other hand, some new features of the latest 802.11 standard may cause adverse affects to the co-existing legacy nodes. For example, the MU-MIMO feature may result in the deafness problem, and QoS prioritization and MAC layer aggregation cause significantly lower performance in classical hidden node scenario - even with the RTS/CTS handshake mechanism. Also, delay and/or packet loss in TCP significantly decreases end-to-end throughput due to congestion control. This thesis studies and verifies these problems, proposes a methodology to mitigate the deafness problem in MU-MIMO, analyzes experimentally the effectiveness of multiple TCP connections to improve TCP throughput, and finally provides a framework via multiple-synced-sniffing methodology to diagnose hidden-node problems.

ÖZETÇE

802.11 ağlar, günlük yaşantımızda önemli yer tutan taşınabilir cihazların yaygınlaşmasını mümkün kılmıştır. Bu ağ ailesinin -şuanki- en gelişkin üyesi 802.11ac, daha yüksek kipleme oranı, uzaysal akım, band genişliği ve uydu-yer bağlantısında kullanılan çok kullanıcılı çok girdili çok çıktılı haberleşme teknikleriyle seleflerinden daha yüksek başarım oranı sağlamaktaıdır. Fiziksel katmandaki bu yeni tekniklerin yanısıra; 802.11ac erişim katmanında veri birleştirilmesi ve önceliklendirilmesini de seleflerinden miras edinmiştir. Yukarıda bahsedilen özelliklerden, çok kullanıcılı çok girdili çok çıktılı sistemler -öncesiyle uyumlu olmadıklarından, 802.11ac protokolünü desteklemeyen düğümlerde sağırlık sorunu oluşturarak; ağdaki başarım oranını düşürmektedir. Benzer şekilde, klasik saklı düğüm topolojisinde de, erişim katmanında uygulanan veri birleştirilmesi ve önceliklendirilmesi; RTS/CTS el sıkışmasının kullanılmasına rağmen ciddi veri kaybına yol açmaktadır. 802.11 ağlardakı bu gibi problemlerin oluşturduğu veri kaybı ve gecikmesi, uçtan uca veri aktarım kontrolü için kullanılan TCP protollerindeki tıkanıklık kontrol algoritmalarından dolayı da -ilave olarak- başarım oranını düsürmektedir. Bu tezde 802.11ac protokolünü desteklemeyen düğümlerde oluşan sağırlık sorununu önleyici yaklaşım, çoklu bağlantılarla uçtan uca TCP başarım oranını arttıcı deneysel çalçışmalar ve saklı düğüm problemini analiz etme metodolojisi sunulmuşur.

ACKNOWLEDGEMENTS

I would firstly like to mention about my first math teacher Fatma Barut from primary school, since she helped me realize joy of learning at my early ages. As a bit restive child, this was quite important for me.

Unfortunately, I did not have that joy during my high-school years and became unwilling about university education; although I applied for Işık University.

There, I had a chance to attend courses from very nice -and from time to time highly amazing- people, including Türker Bıyıkoğlu, Esin İnan, Hilmi Demiray, Erinç Özden, Mark Shields, Ali İnan, Onur Kaya, and Umut Ekmekçi. The last but one inspired me to study/work in the field of communications, which induced this thesis.

My next step was Özyeğin University. Here, I participated into collaboration of university with ICT manufacturer AirTies Wireless Networks and Ministry of Science, Industry and Technology of Turkey, which funded this thesis under SANTEZ projects 01621.STZ.2012-2 and 0231.STZ.2013-1. Here, I experienced joy, dissatisfaction, failure, and success of research and development for the first time, under supervision of Oğuz Sunay and Ali Özer Ercan. Beside my research, professors in Özyeğin University was also nice and had high level of enthusiasm. I increased my knowledge via attending courses of Murat Uysal, Okan Örsan Özener, Ahmed Akgiray, Gonca Gürsun, Reha Civanlar, Tanju Erdem, and Barış Aktemur. TAs in Özyeğin University were also great for my graduate life. I would like to express my thanks to Volkan Yazıcı, Ali Arsal, Murat Kırtay, and Kyoomars Noghani for their support and many others for their friendship.

Before writing this thesis (and even completing all stuff of it), I've joined a group of another nice people that constitutes Technology Research department of AirTies

Wireless Networks. Here I had many discussions on design and implementation with great folks including Okan Palaz, Maximilian Fricke, Çağrı Sofuoğlu, Mustafa Karaca, Sarper Göktürk, and İrfan Acar. My former colleague Mehmet Karaca deserves a special thank for his restless work and positive attitude during entire experimental investigations. I would also like to mention on my former director Eren Soyak for being an exemplifying person, who truly likes toughness instead of getting concern.

Here, I would like to dedicate the last paragraph for three people. Two of them knows me since my birth and gave a birth to third one upon my kind request, when I was four. These three had always supported me and had a respect on my decisions -even after long debates, without any uncertainty nor expectations. I've always admired their true support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In 1985, the Federal Communications Commission (FCC) released the spectrum from 2.4-2.5 GHz for use by the Industrial, Scientific and Medical communities. This exciting news meant that the spectrum would be available for developers of wireless communication technologies without requirement of licensing fees; led many developments that were far from today's ubiquitous, sprawling networks.

In the early 1990s, however, the IEEE realized that a wireless communications infrastructure was necessary to meet a clearly desirable market niche and established an executive committee -as part of the IEEE 802[1] standard, to focus on developing a wireless LAN (Local Area Network) standard [1]. This aim introduced "11" family of IEEE 802 standards (IEEE 802.11), aka Wi-Fi. At that time, there were no handheld mobile phones that utilized Wi-Fi and very few laptops. The challenge was providing a reliable, fast, inexpensive wireless solution in 2.4-2.5 GHz band that could grow into a standard with widespread acceptance.

In 1997, the IEEE ratified the original *802.11 standard* with a maximum data rate 2 Mbps, which holds compatibility with wired Ethernet networks in MAC layer -see Figure 1. From that day to this, 802.11 networks has enabled and is fueled by the proliferation of connected mobile devices with a wide variety of applications that have been integral parts of our daily lives.

In this thesis, we first dedicate this chapter to introduce working principles of 802.11 protocol. Following this, we provide two throughput reductive case studies that grew up from evolution of the protocol in Chapters 2 and 3 with our proposed

[1]The general IEEE designation for network standards

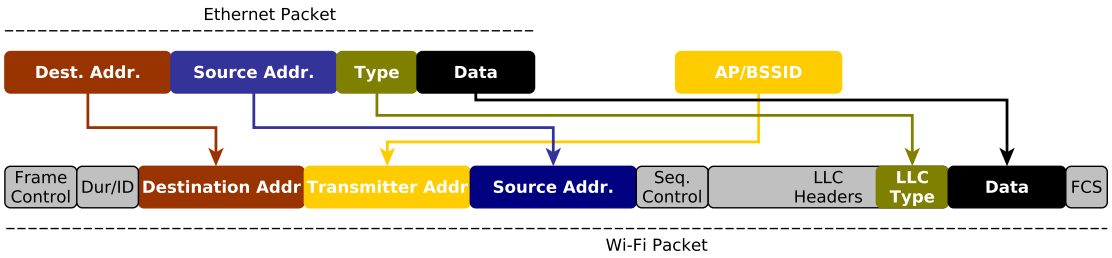

Figure 1: 802.3 (ethernet) to 802.11 (Wi-Fi) encapsulation.

solutions. Lastly, we consider cross-layer optimization of Wi-Fi with TCP in Chapter 4 and draw conclusions in Chapter 5.

1.1 IEEE 802.11 summary

In this subsection, we would explain how does the client connect to the access point, how do they operate and which problems do they experience during the operation time.

1.1.1 Establishing Connection

Access points bridge traffic in between mobile stations and other devices on the network. Before streaming data on the network, a mobile station should be in appropriate connection state (authenticated and associated) with an AP.

The three 802.11 connection states are:

- Not authenticated or associated
- Authenticated but not yet associated
- Authenticated and associated

In order to join a Wi-Fi network, a mobile station first sends a `probe requests` to discover 802.11 networks within his proximity. Probe requests advertise his supported data rates and 802.11 capabilities.

After receiving a probe request, an AP -or APs- would try to find at least one common supported data rate with the mobile station. If there is, an AP would send a `probe response`, which advertises his SSID[2], supported data rates, encryption types and other 802.11 capabilities.

Consequently, the mobile station chooses compatible networks from probe responses that he received. Once compatible networks are discovered, the mobile station will attempt to authenticate by sending 802.11 authentication frame with sequence number 0x0001.

After receiving the authentication frame, the AP responds with authentication frame set to open indicating a sequence of 0x0002. At this point the mobile station is authenticated but not yet associated. If an AP receives any frame other than an authentication or probe request from the mobile station, that is not authenticated, it will respond with a de-authentication frame; placing the mobile into an unauthenticated and non-associated state.

Once the mobile station determines an AP to associate, he will send an association request. This frame contains chosen encryption types if required and other compatible 802.11 capabilities.

If the elements in the association request would match the capabilities of the AP, the AP will create an Association ID for the mobile station and respond with an association response with a success message - this grants network access to the mobile station. Now the mobile station is successfully associated to the AP and data transfer can begin -see Figure 2.

[2]Service Set Identifier, wireless network name

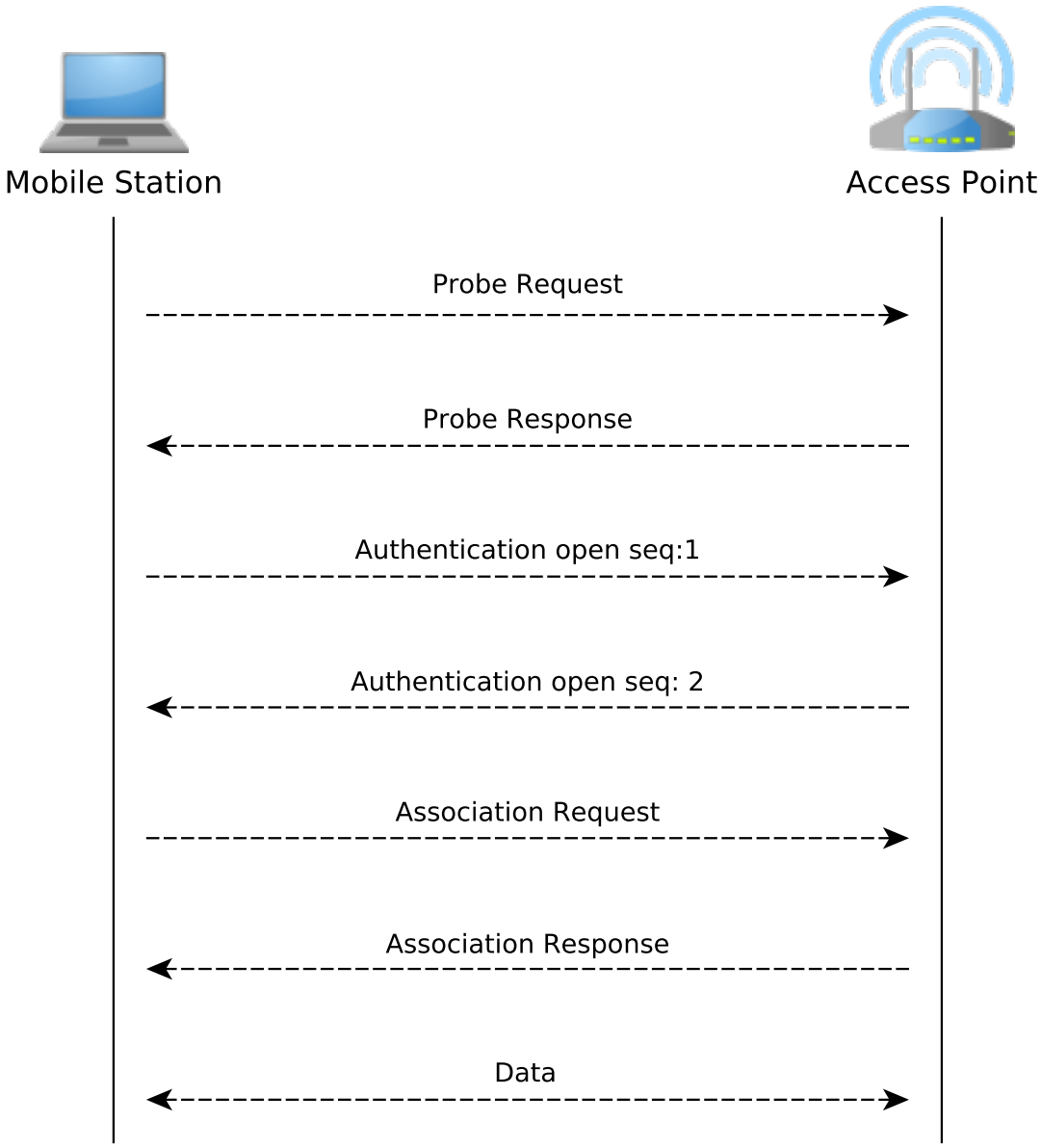

Figure 2: Association Procedure of mobile station to AP

1.1.2 Data Transfer in 802.11

Once the mobile station connects to the AP, he utilize the bandwidth with respect to the distributed or coordinated scheduling, which is selected by AP from the options below.

- Point Coordination Function (PCF)
- Distributed Coordination Function (DCF)

The PCF has not been embedded in most of 802.11 chipsets and all of the work in this thesis is considered with the existence of DCF. Nevertheless, we would dedicate next paragraph to shortly explain PCF.

The point coordination function allows an 802.11 network to provide an enforced "fair" access to the medium where access to the medium is restricted by the AP. At the beginning, the AP transmits a `Beacon` frame containing the `CF`[3] `Parameter Set` element to gain control of the wireless medium. Consequently, associated stations do not contend for the medium and can transmit data only when they are allowed (actually polled) by the AP. Even though PCF has been part of the 802.11 standard from the beginning, due to it's higher overhead and interoperability (in between different vendors) problems, vendors have always been reluctant to activate it.

Unlike PCF, DCF is a mechanism where mobile stations have channel access with contention. Nevertheless, 802.11 APs are not able to decode concurrent transmissions on same frequency. So, if more than one mobile station would transmit data towards an AP on same time and frequency, none of them could utilize the network. For this reason, each station with a packet to transmit, has to be sure that nobody is using channel[4] for a constant amount of time[5]. This is done by simply measuring the amount

[3] `Contention Free`
[4] Wi-Fi is usually called as listen-before-talk protocol for this reason
[5] DIFS, distributed inter-frame space

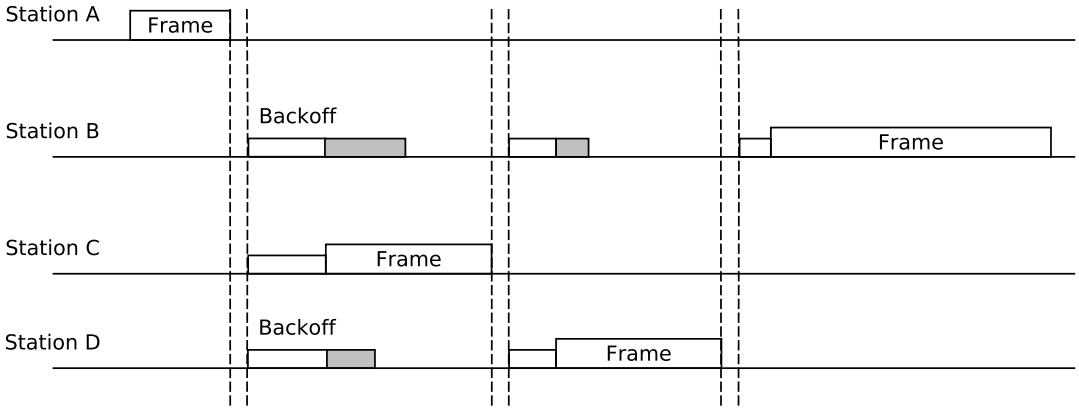

Figure 3: Illustration of Carrier Sense Multiple Access in 802.11 networks.

of energy received on channel -called `Carrier Sensing`. Initiating transmission just after this constant amount of time would still cause concurrent transmissions -which is called *collision*. For this reason, stations initialize a back-off timer with a randomly selected duration, decrement it every time they sense an idle channel, then initiate a packet transmission.

This mechanism could be followed from Figure 3. Stations B, C, and D have packets to transmit, while Station A has data transmission. After completion of Station A's transmission and constant amount of time, all of them obtains a random back-off duration in between $[0, CW)$[6]. Since Station C obtained lower back-off duration, he consumes it and initiates his transmission earlier than others. Following this, Stations B and C repeat the contention procedure.

The final task of the transmitter is waiting for an acknowledgement. If his transmission was successful, the receiver would respond with an `ACK` frame and if the transmitter would also receives an `ACK` frame successfully, he would be able to decrease his current `Contention Window` to minimum value of existing traffic[7]. This gives an earlier access opportunity for the next transmission. As represented in the

[6]CW: Contention Window

[7]We would provide more details on it in next subsection

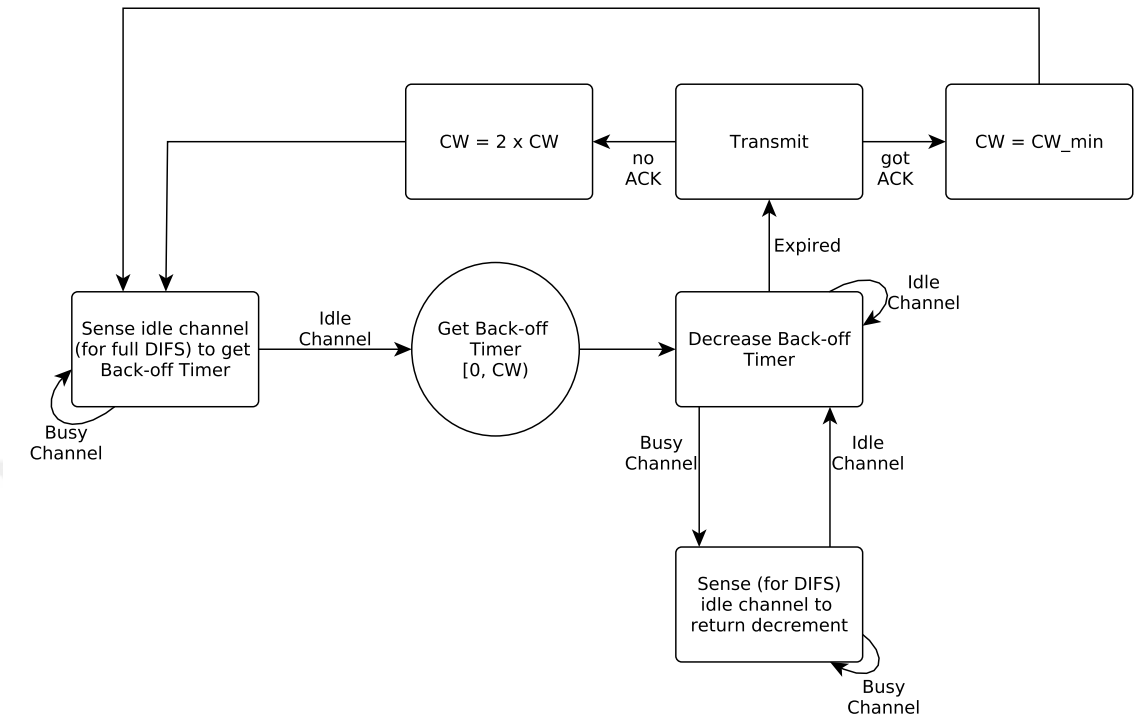

Figure 4: Flow Diagram of 802.11 Carrier Sense Multiple Access.

Figure 4, upon unsuccessful data transmission or `ACK` reception, transmitter doubles his `Contention Window`.

Due to the distance and existence of various signal fading elements in between nodes (such as walls), it is possible for one station to not sense transmission of the other. In such case, a mobile station could initiate a concurrent transmission with his neighbour, which causes a *collision* -see Figure 5. This phenomenon is called the *hidden node problem* which significantly decreases network performance due to waste of time.

IEEE 802.11 has an optional `RTS/CTS`[8] mechanism to mitigate the effect of this phenomenon. In this case, after the back-off timer, the transmitter tries to reserve the medium until end of expected `ACK`[9] frame. This is expressed via `Duration` (2 bytes)

[8] `Request to Transmit / Clear to Transmit`
[9] `Acknowledgement`

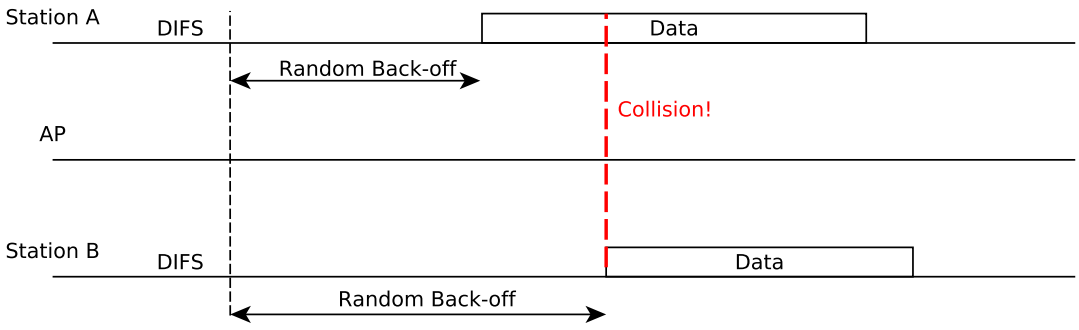

Figure 5: Illustration of a collision caused by a hidden node setting: Station B consumes back-off timer during transmission of Station A, since he does not sense; consequently starts transmission towards AP too.

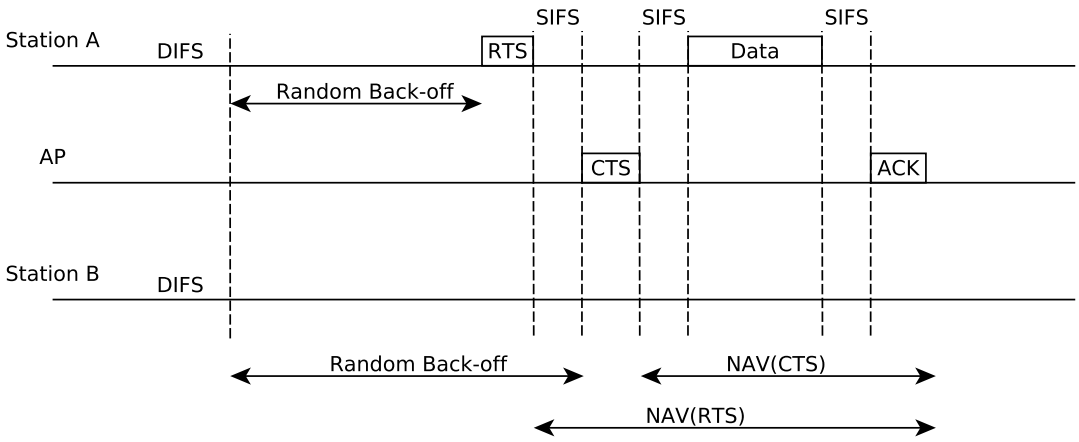

Figure 6: Illustration of RTS/CTS control frame exchange.

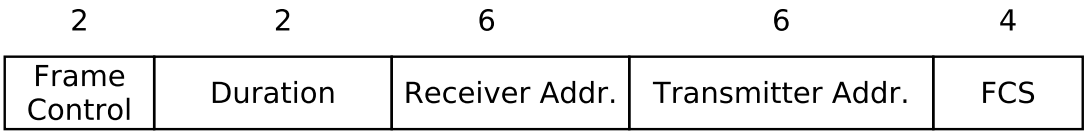

Figure 7: RTS frame format.

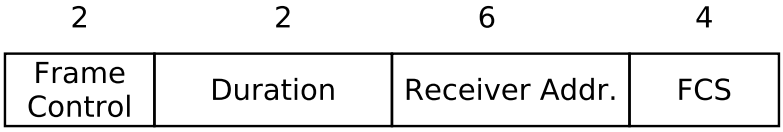

Figure 8: CTS frame format.

field of `RTS` frame -see Figure 7. If the receiver successfully receives and decodes the `RTS` frame, he responds with `CTS` frame, which also contains `Duration`[10] field -see Figure 8. Thus, *hidden* node decodes one of these two frames and does not access the channel for a said duration.

1.2 Improvements of IEEE 802.11 in History

In this subsection, we present some of major improvements of 802.11 in physical layer and medium access control layer.

1.2.1 PHY Layer Improvements

As it could be observed from Table 1, maximum link rate of 802.11 increased from 2 Mbps to 6.9 Gbps in 19 years. This improvement is mostly based on following PHY

[10]Note that, the value of the `Duration` is updated, since aim is reserving medium until end of the `ACK` frame.

layer elements: Modulation and Coding Scheme (MCS), Bandwidth, Multiple Input Multiple Output communications.

From the original 802.11 to today's 802.11ac, Modulation and Coding Scheme increased from $\frac{1}{11}$ QPSK to $\frac{5}{6}$ 256 QAM, bandwidth is increased from 20 MHz to 160 MHz and communication with 8 spatial streams were enabled.

802.11 prot.	Year	Freq. (GHz)	Max. BW (MHz)	MIMO	MU-MIMO	Max. MCS	Rate (Mbps)
802.11	1997	2.4	22	✗	✗	QPSK × 1/11	2
a	1999	3.7/5	20	✗	✗	QAM-64 × 3/4	54
b	1999	2.4	22	✗	✗	QPSK × 1/2	11
g	2003	2.4	20	✗	✗	QAM-64 × 3/4	54
n	2009	2.4/5	20, 40	4	✗	QAM-64 × 5/6	600
ac	2013	5	160	8	4	QAM-256 × 5/6	6930

Table 1: IEEE 802.11 network standards PHY information

In addition, multiple user transmission on same frequency and time is firstly implemented with release of IEEE 802.11ac

Beyond the deployed standards in Table 1, IEEE 802.11 has ongoing developments in following protocols for particular purposes.

- **802.11ah (Wi-Fi HaLow)**[11]: Operates in sub 1 GHz license-exempt bands. The aim is providing lower energy consumption and higher range. Allows offloading cellphone tower traffic and creation of large groups of stations (or sensors) that cooperate to share the signal, supporting the concept of the Internet of Things (IoT) [2].

- **802.11ad and 802.11ay**: Operates in 60 GHz band which has significantly different propagation characteristics than 2.4 GHz and 5 GHz.

- **802.11ax**: Operates in 5 GHz band, supports 4×4 MIMO where each spatial stream and 2 MHz sub-carrier could be scheduled via OFDMA. The aim is

[11] "HAY-Low"

providing a fair and coordinated scheduling in dense WLAN scenarios –such as stadiums or trains [3].

1.2.2 MAC Layer Improvements

Besides aforementioned improvements in PHY layer above, the evolution of 802.11's MAC layer also provided significant gains in terms of throughput and lower energy consumption. Below, we provide a detailed information on major ones.

802.11 prot.	Purpose
e	Enhancements; QoS including packet bursting
f	Inter-Access Point Protocol (IAPP)
h	5 GHz spectrum, Dynamic Channel/Frequency Selection
k	Radio resource measurements
r	Fast roaming
s	Wireless mesh networking
u	Interworking with non-802 networks[12]
v	Wireless network management

Table 2: IEEE 802.11 amendments

1.2.2.1 Frame Aggregation

802.11n was proposing extremely higher throughput (with MIMO and wider bandwidth) than his predecessors and there was less tolerance for existing inter-frame time overhead in between successive transmissions. For this reason, frame aggregation is introduced with this protocol.

As we mentioned above, transmitter node waits an acknowledgement after transmission. In detail, this "data transmission" event has following overhead[13]:

- Constant wait (DIFS[14])

[13]-See Figure 9

[14]Distributed Inter Frame Space time, usually 34 us

- Back-off wait. For best-effort traffic; CW is uniformly selected from $[0, 2^4)$ in best case and $[0, 2^{10})$ in worst case, in terms of slot time[15].
- SIFS[16] wait. Wi-Fi chips require this time to change mode from transmitter to receiver (or vice versa)

Here, we provide A-MPDU[17] methodology, which is most commonly used approach for frame aggregation in today's Wi-Fi chips. Each 802.11 packet (as in Figure 1) is Mac Protocol Data Unit. Putting them back-to-back with MPDU delimiter (to declare beginning and length of MPDU unit) would create an A-MPDU frame. So, to avoid an inter-frame overhead, transmitter might transmit a single A-MPDU frame and receiver would response with a Block Acknowledgement (which contains sequence start number and bitmap) to inform which of the transmitted sequences are received correctly. Since each MPDU has its own FCS[18], receiver is be able to decode and acknowledge them separately/independently.

1.2.2.2 Quality of Service

In order to differentiate delay-sensitive applications, such as Voice over Wireless LAN and multimedia streaming, IEEE released the 802.11e amendment in the context of *MAC Enhancements QoS*. The purpose of the amendment was providing a higher chance of channel access for higher priority traffic -which is usually provided in `Type of Service (ToS)` field of IP header. This is accomplished via setting different minimum and maximum Contention Window values for different types of traffic (usually called as Access Categories) as illustrated in Table 3.

[15] Mostly 9 microseconds
[16] Short Inter-Frame Space time, usually 16 microseconds
[17] Aggregated-Mac Protocol Data Unit
[18] Frame Check Sequence

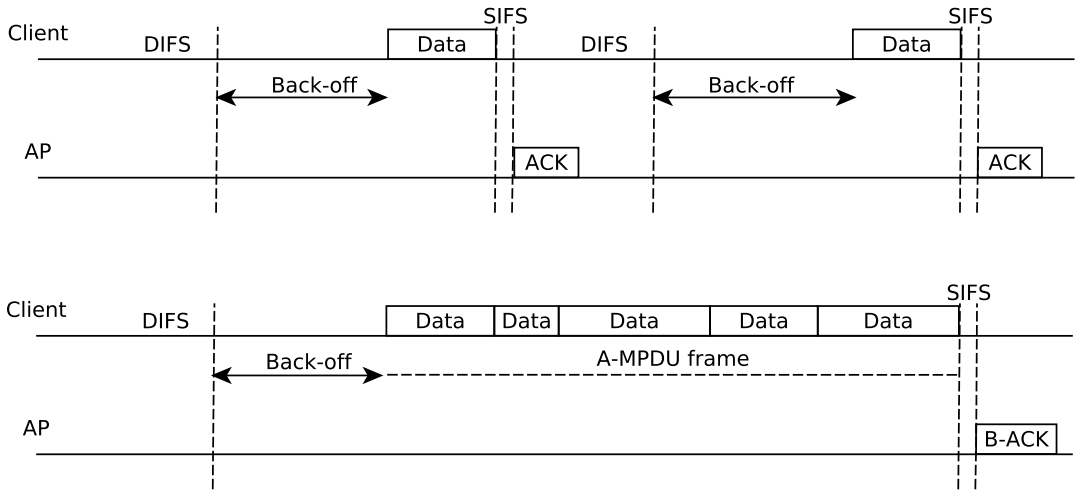

Figure 9: Representation of frame aggregation.

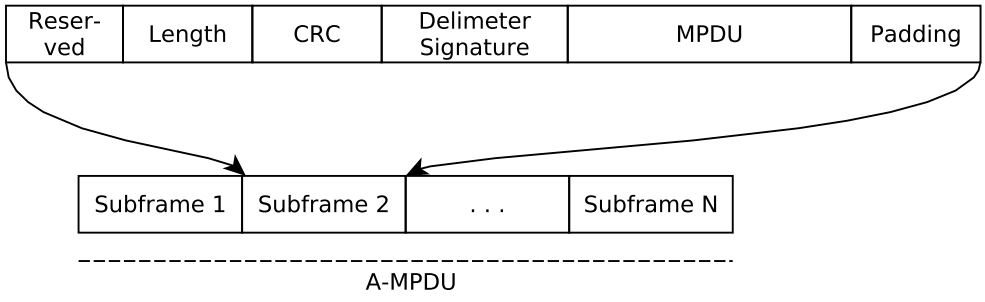

Figure 10: Construction of A-MPDU packet.

Access Category	CW_{min}	CW_{max}
Background (AC_BK)	15	1023
Best Effort (AC_BE)	15	1023
Video (AC_VI)	7	15
Voice (AC_VO)	3	7
Legacy DCF	15	1023

Table 3: Default CW values in 802.11e

In Equations 2 and 1 we provide the average time of channel access for video traffic and best effort traffic, when there is no unsuccessful transmission.

$$AC\_VI_{avg} = \texttt{DIFS} + \frac{7}{2} \times \texttt{SLOT} = 65.5\mu s \tag{1}$$

$$AC\_BE_{avg} = \texttt{DIFS} + \frac{15}{2} \times \texttt{SLOT} = 101.5\mu s \tag{2}$$

1.2.2.3 Wireless Mesh Networks and Roaming

IEEE 802.11 networks are able to construct wireless backbone to increase coverage and/or throughput with usage of Wireless Distribution System (WDS). Routing traffic flow in mesh network, hand-off connectivity of client from AP to AP and band steering (such as forcing station to use 5 GHz, instead of using interference existing 2.4 GHz) of mobile station has always been though challenge -and research interest- for industrial and academic communities in such systems.

In order to ease hand-off operation, IEEE 802.11k is released in 2008 which provides topology information messaging in between AP and client. Another relevant amendment for hand-off operation is IEEE 802.11r, which gives a chance to AP to send "roam" directive to client.

CHAPTER II

EXPERIMENTAL INVESTIGATION OF THE IMPACT ON TCP TRAFFIC BY HIDDEN VIDEO-TAGGED UDP TRAFFIC

In this chapter, we investigated the impact of video-tagged UDP traffic on best-effort TCP traffic in an experimental setup with a novel observation approach [4]. Our setup consists of three 802.11ac nodes in a hidden node topology, where a 20 Mbps UDP and a best effort TCP traffic flow from the edge (hidden) nodes to the center node, as illustrated in Figure 11.

We observe that, even though the RTS/CTS mechanism is enabled, when UDP and TCP traffic flow simultaneously, video-tagged UDP traffic causes the best-effort TCP traffic throughput to drop by 50%.

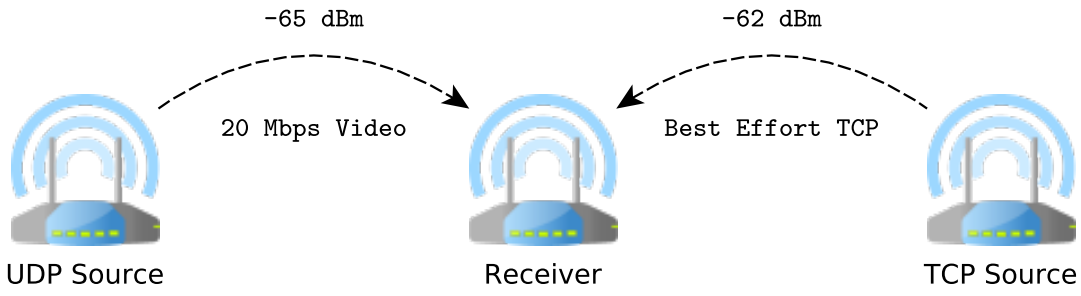

Figure 11: Graphical illustration of investigated hidden node problem.

Below, we first present our investigation approach of the case and causes that we extracted.

2.1 Investigation Methodology

First, we initiated TCP (best effort) and UDP (20 Mbps) traffic towards the same receiver from different nodes via `iperf`[1] for 60 seconds. Afterwards, we repeated the experiment with "video tagging" UDP traffic. Here, we used `iperf`'s tagging support, which actually manipulates `Type of Service (ToS)` field of IP header. By this way, AP uses different minimum/maximum `Contention Window` values, as shown in Table 3.

Video Tagging	TCP	UDP	$\text{UDP}_{\text{jitter}}$	UDP_{lost}	$\text{UDP}_{\text{total}}$
✗	270 Mbps	20 Mbps	0.67 ms	65	108664
✓	129 Mbps	20 Mbps	0.57 ms	591	224572

Table 4: Throughput, delay, and packet loss statistics of video tagged and non video tagged UDP traffic with existence of best-effort TCP.

As it could be seen from Table 4, IEEE 802.11e's default "video tagging" compliance drastically decreases TCP throughput in this topology.

Video Tagging	$\text{TCP}_{\text{tx\_rts}}$	$\text{TCP}_{\text{rx\_cts}}$	$\text{UDP}_{\text{tx\_rts}}$	$\text{UDP}_{\text{rx\_cts}}$
✗	38896	27570	25088	16379
✓	42381	17215	113661	50603

Table 5: Counters of transmitted `RTS` from sources (`TCP`, `UDP`) and received `CTS` frames that addressed to sources.

Consequently, we obtained number of transmitted RTS frames from TCP and UDP sources[2] and number of received CTS frames (that addressed to this sources) via sources -see Table 5. We suprisingly realized that, UDP source is transmitting RTS frames 4 times more when video tagging is enabled and half of those frames are not replied by receiver.

Collision of RTS frames from different sources is acceptable, since RTS is a control frame that has less air-time usage than actual data. Also, after such case, collision

[1] Frequently used network bandwidth measurement tool[5]

[2] Using (`wl -i wl1 counters`) command of Broadcom 4360 Chip

experienced node updates Contention Window, whereas this collision probability decreases. Nevertheless, such drastic throughput degradation is not expected at with RTS collisions of hidden nodes.

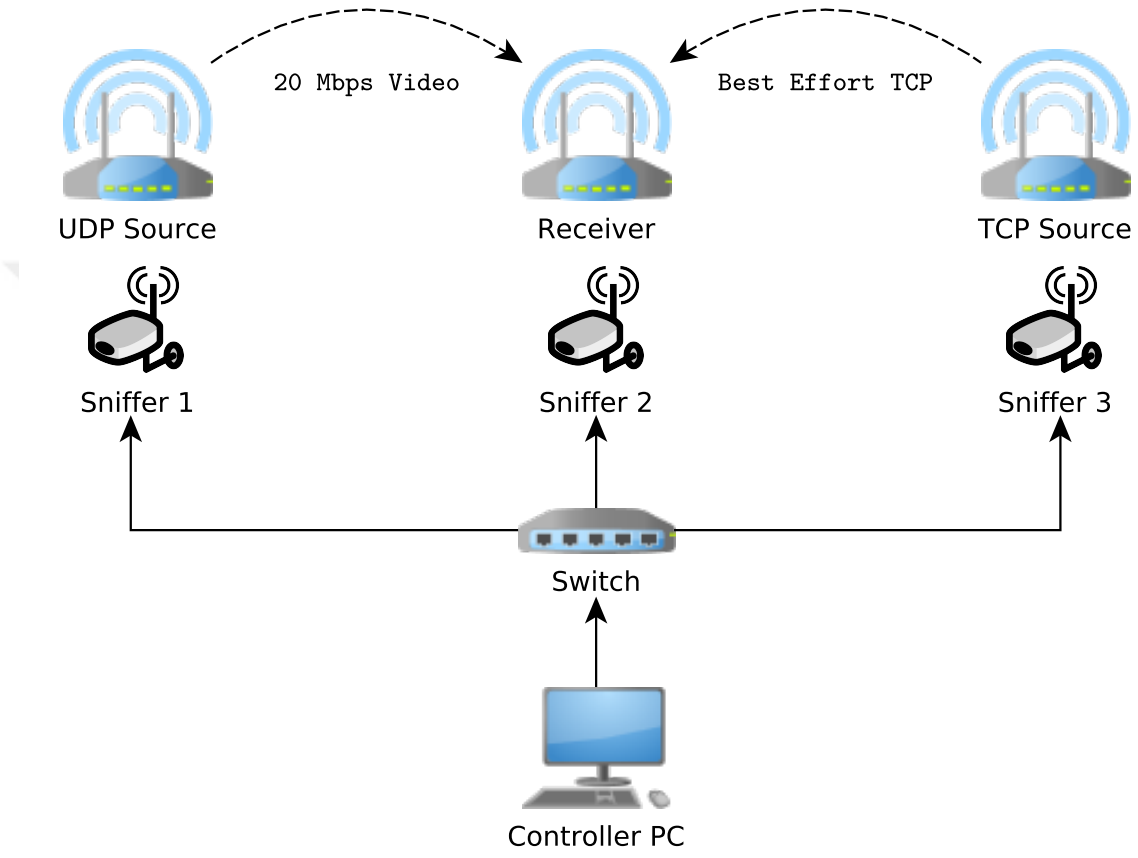

Figure 12: Constructed sniffer setup to investigate problem.

At this stage, we aimed to understand the problem and started development of a tool that would just plot the data on the air, as in previous Figures 5 or 6. We used Python programming language for back-end, Javascript for front-end development.

First, to gather packets on the air, we put sniffer devices[3] in the vicinity of all three devices in network and sniffed the medium via `tcpdump`[6]. Consequently, we parsed the data via open source *libpcap* parsing library `pypcapfile`[7] in Python -contributed to it by implementing parsing of Wi-Fi packets.

[3]We used `Qualcomm QCA9980 (Beeliner)` cards for this purpose

Wireless drivers provide each of the Layer 2 packets in an A-MPDU block as separate packets. Nevertheless, these packets are transmitted as a whole in Layer 1; thus, we need to re-assemble Layer 2 packets before air-time calculation of A-MPDU blocks. Most wireless drivers timestamp all packets within an A-MPDU block with the same timestamp (`radiotap.mactime`). We used this to group A-MPDU packets together.

Consequently, we calculated air-time of each packet, considering physical layer parameters -such as modulation and coding scheme, bandwidth, number of spatial streams and guard interval. This information is available at `radiotap` [8] headers in captured files, which is produced by wireless drivers too.

The next task was synchronization in the granularity of microseconds; since we want to relate captured frames from different sniffers. Here, all sniffers are able to hear transmissions of receiver and the source in his vicinity.

At this point, we decided to use timestamps of periodically (usually in 100 milliseconds) transmitted `Beacon` frames from receiver node, since it's timestamp is broadcasted on the air and unique. Nevertheless, we realized that clock speed of processors of sniffing devices are not same, $1\mu s$ to $2\mu s$ shift is possible in each 100 milliseconds. Here we realized that sync in 100 milliseconds won't provide enough granularity to detect RTS collisions.

In this case, we found that common Block Acknowledgement frames that are transmitted from receiver might be useful. In all experiment period, B-ACKs can be quite repetitive. However, considering B-ACKs jointly with Beacons provide enough uniqueness to sync timestamps. Our approach was as following:

- Find a beacon frame that received from all three sniffers.
- Find the first B-ACK frame that is transmitted after common beacon and received from all sniffers.

- Sync timestamps and repeat the previous step

Distinguishing B-ACK frames are relatively easy in limited interval, since they contain sequence number information which rounds in $[0, 4096)$. Thus, we synced timestamps of captured packets in granularity of microseconds.

2.2 Root Cause: RTS_CTS collision

After calculating the air-time of each sniffed packet and synchronizing timestamps of captured packets from multiple devices, we used Mike Bostock's `D3.js` library to visualize our findings[9].

Here, we visually observed the following pattern that we express step-by-step below (also represented in Figure 13).

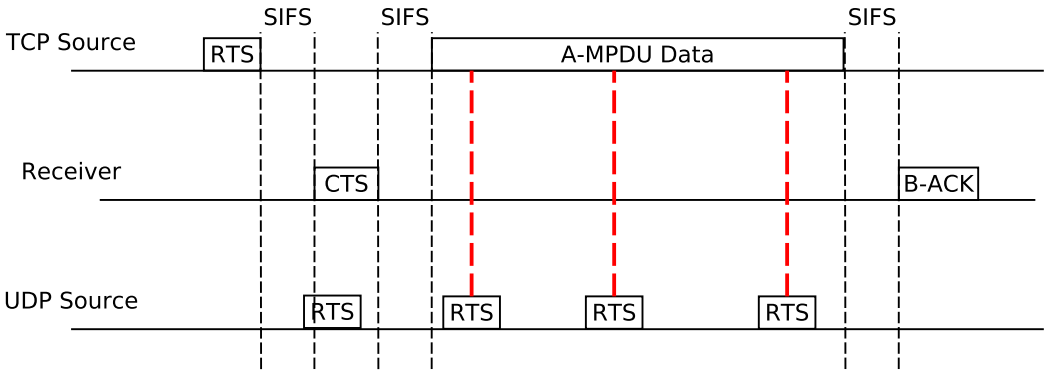

Figure 13: Graphical illusturation of root cause. UDP transmitter initiates RTS frame concurrently with receiver's CTS (going towards TCP source). Consequently, A-MPDU block is consecutively collided with RTS frame of UDP source.

1. TCP source initiates an RTS frame towards the receiver, then waits for a SIFS duration.

2. UDP source is hidden, did not sense this RTS frame, hence initiates a RTS frame towards the receiver too.

3. Receiver decodes the RTS of the TCP source and starts to switch from receiving mode of hardware to transmitting mode –in SIFS period. Therefore, the receiver could not decode (even not sense) RTS frame of the UDP source.

4. Receiver initiates a CTS frame towards the TCP source. Since TCP source does not hear transmission of the UDP source, he can successively decode this CTS frame.

5. After receiving the CTS frame, TCP source initiates an aggregated data frame.

6. UDP source does not get any response for his RTS frame, has an idle wait and initiates another RTS frame, which causes collision with A-MPDU block. UDP source repeats this procedure until the end of A-MPDU.

7. Receiver decodes some of the MPDU blocks in A-MPDU and acknowledges the TCP source.

8. TCP source gets a B-ACK, even his transmission was partially unsuccessful, does not increase his Contention Window.

The effect of video tagging becomes very critical at Step 6, since Contention Window minimum/maximum limits in video access category are low. For this reason, when UDP traffic is video tagged, more frequent RTS frames collides with more MPDUs, hence drastic degradation occurs in TCP traffic.

In order to measure the negative impact, we compared transmitted MPDU's via the sequence number field[4] of data packets and acknowledged MPDU's via the starting sequence number[5] and the bitmap[6] field of Block Acknowledgement packet from capture of sniffer next to TCP source. Here, we provide a histogram of failed MPDU's in Figures 14, 15. As it could be realized, "video tagging" causes more failed MPDU's.

[4] `wlan.seq`
[5] `wlan_mgt.fixed.ssc.sequence`
[6] `wlan.ba.bm`

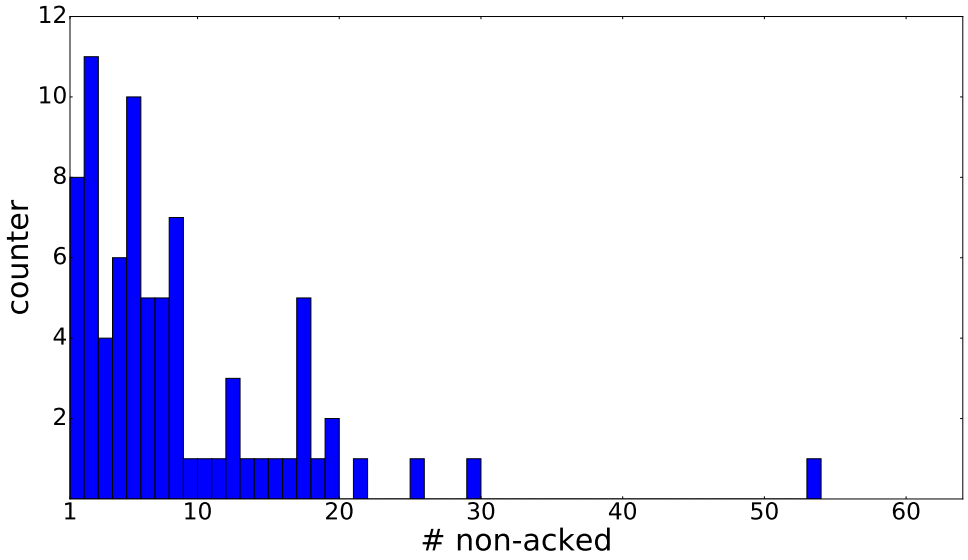

Figure 14: Histogram of failed MPDU's (in 10 seconds), jointly extracted from Block Acknowledgement and A-MPDU packets, in TCP traffic when UDP traffic is best effort.

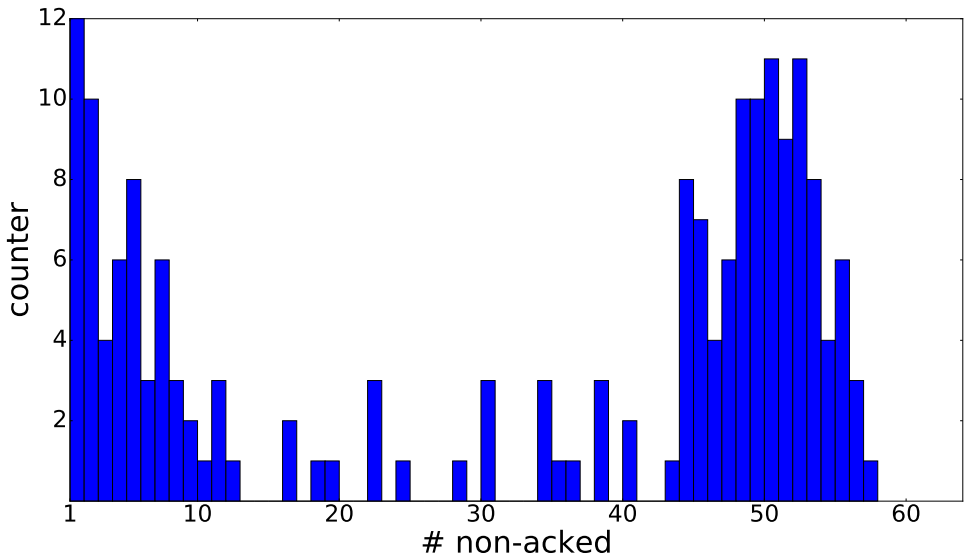

Figure 15: Histogram of failed MPDU's (in 10 seconds), jointly extracted from Block Acknowledgement and A-MPDU packets, in TCP traffic when UDP traffic is video tagged.

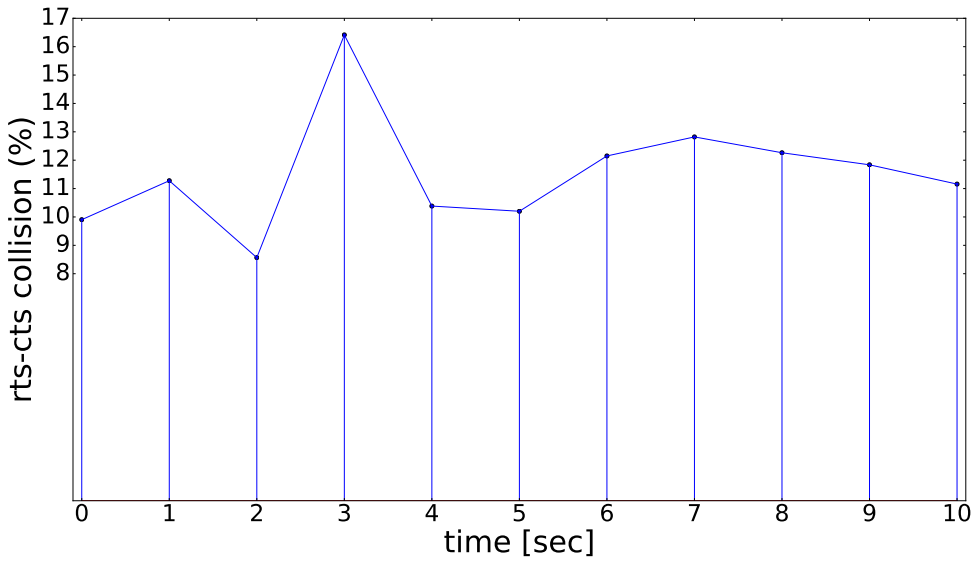

Figure 16: Ratio of RTS packets (in 10 seconds) from video tagged UDP source that collided with CTS packets from receiver over successful RTS packets.

2.2.1 Probability of RTS_CTS collision

Although occurrence of initial RTS-CTS collision event seems to be very rare, we found the probability of it as %11.58 during our studies. Here, we extract number of successful RTS transmissions from the UDP source, then count overlaps of RTS packets from the UDP source and CTS packets from the TCP source through time. Consequently, we calculated the ratio of overlapping RTS-CTS packets and successful RTS packets; plot it on Figure 16.

A simple approximate computation of the collision probability is as follows. In our case, RTS packets from the UDP source can only collide with CTS packets of the receiver, if it's just transmitted in $[0, \texttt{SIFS})$ duration before the target CTS transmission time of the receiver.[7] By using `SLOT`, `SIFS` duration and contention window (for video-tagged node) parameters in Table 3, we can formulate the RTS-CTS collision probability (given that first RTS frame is initiated by TCP source) as following:

[7]Since `SIFS` duration exists for changing hardware mode and the receiver does not listen for any further packets upon receiving the RTS packet from the TCP source.

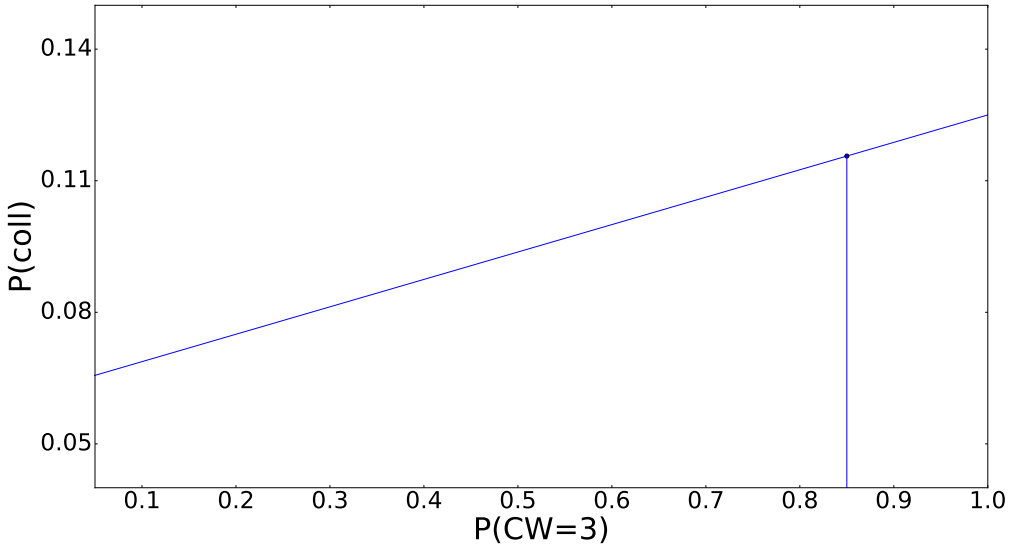

Figure 17: Swap of RTS_CTS collision probability with respect to contention window of video-tagged node.

$$P(coll) = P(coll|CW_3)P(CW_3) + P(coll|CW_4)P(CW_4)$$
$$= \frac{1}{8} \times P(CW_3) + \frac{1}{16} \times P(CW_4)$$

Video tagging provides quicker channel access, since Contention Window limits are lower -see Table 3. This introduces smaller A-MPDU's, which is good for smaller jitter; yet causes more control packet transmission (RTS, CTS and Block Acknowledgement) and inter-frame spacing overhead. Here, we provide histograms in Figures 18, 19, when video tagging is enabled and not.

Shortly after failed RTS frames, we observed that UDP source lowers MCS rate to it's minimum (MCS 0), upon getting the first CTS frame from the AP; although RSSI level is satisfactory for MCS 4. This results in near $\frac{1}{6}$ slower transmissions in UDP traffic -see Table 8. Here, we provide MCS histogram of UDP source when UDP traffic is video tagged or not in Figures 20, 21.

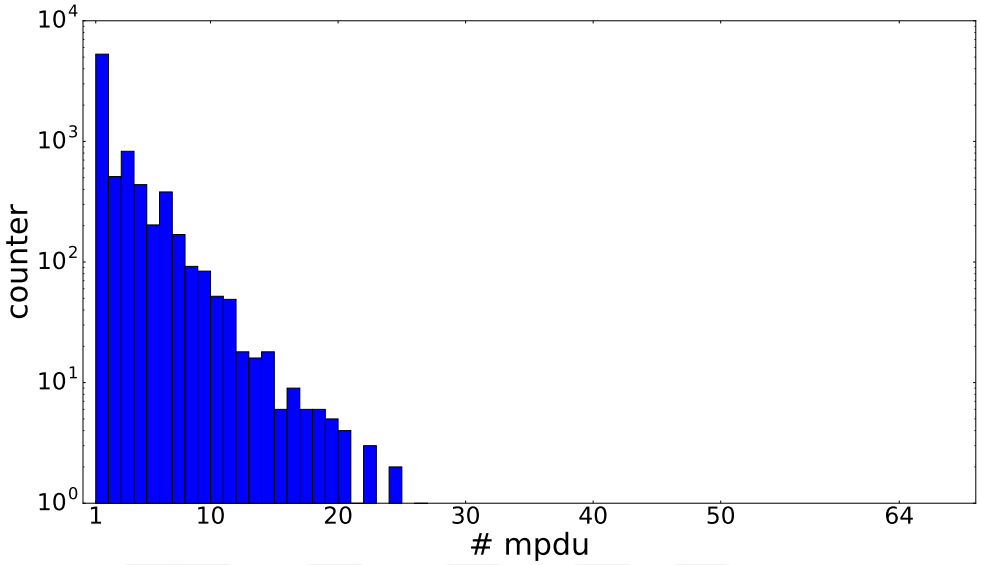

Figure 18: Histogram of MPDU counters (in 10 seconds) in A-MPDU packets transmitted from UDP source when traffic is video tagged

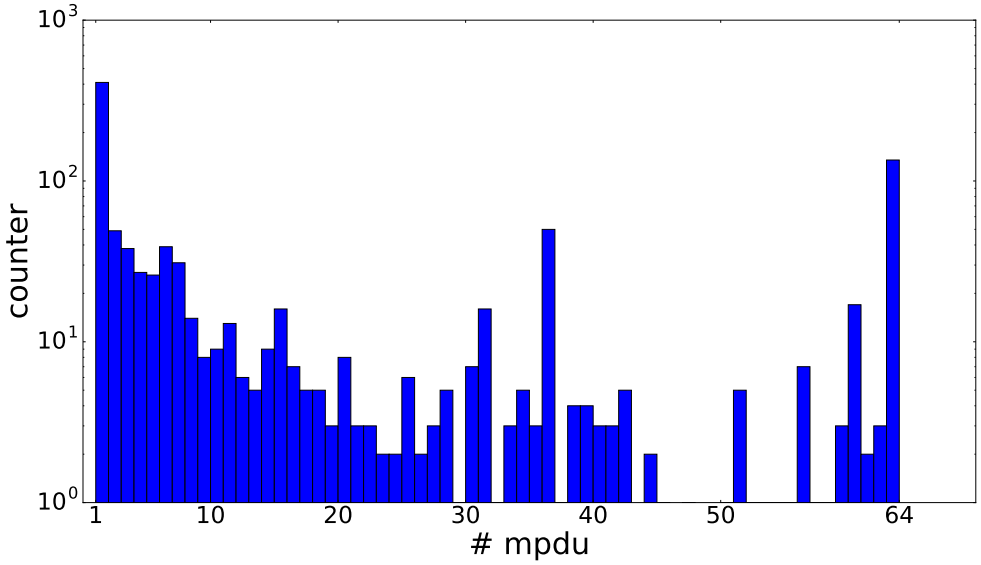

Figure 19: Histogram of MPDU counters (in 10 seconds) in A-MPDU packets transmitted from UDP source when traffic is best-effort

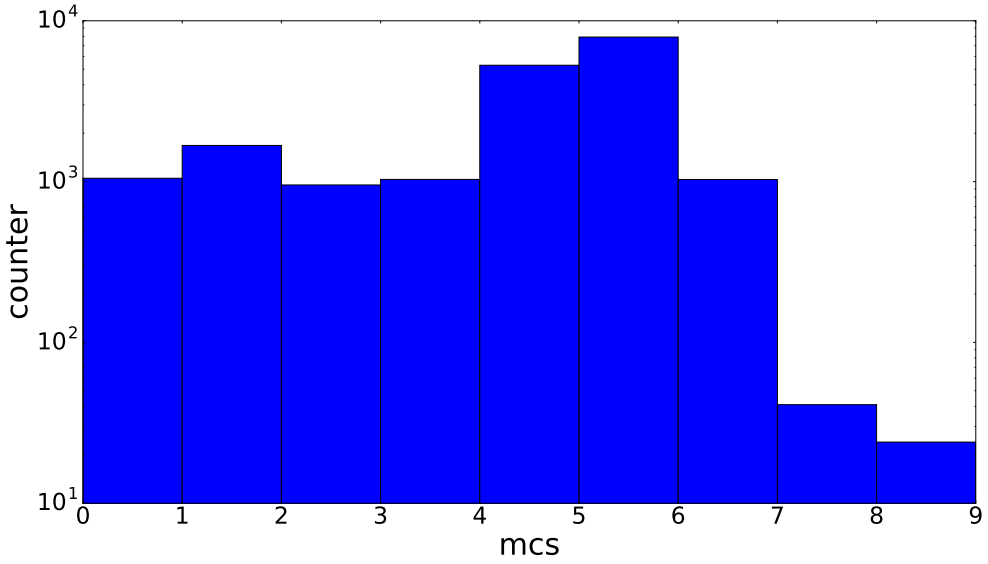

Figure 20: Histogram of MCS counters (in 10 seconds) in A-MPDU packets transmitted from UDP source when traffic is video tagged

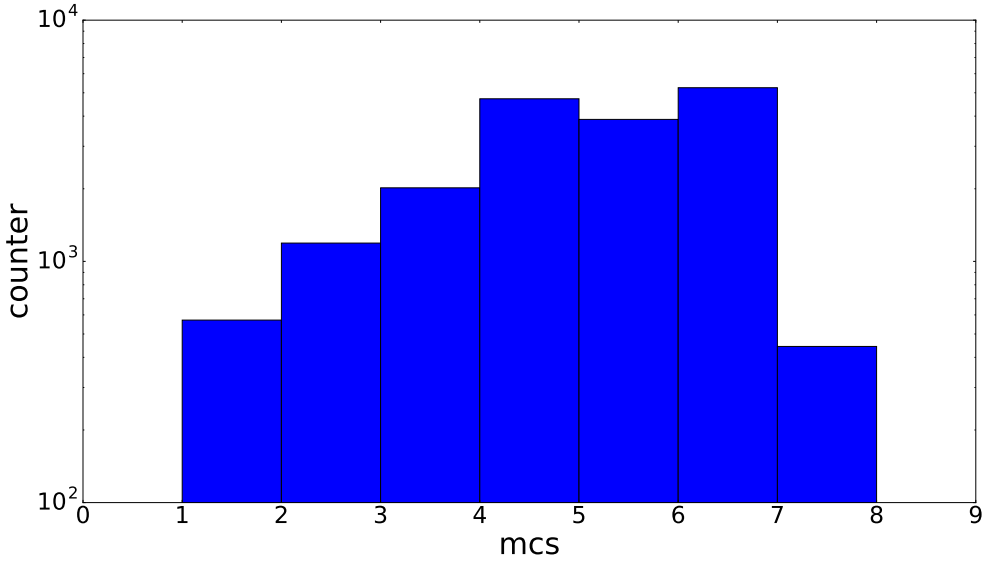

Figure 21: Histogram of MCS counters (in 10 seconds) in A-MPDU packets transmitted from UDP source when traffic is best-effort

2.3 CW Tuning to Mitigate the Effect

In order to protect MPDU frames from consecutive and frequent RTS collisions, we intend to increase inter-arrival times of RTS frames. For this reason, we manipulated minimum/maximum contention window values of video tagged UDP traffic[8]. Below, we present throughput statistics in Table 6 and RTS/CTS statistics of sources in Table 7.

UDP_{CW_MIN}	UDP_{CW_MAX}	TCP	UDP	UDP_{jitter}	UDP_{lost}	UDP_{total}
15	31	149.7 Mbps	20 Mbps	0.579 ms	73	207680
31	63	210.5 Mbps	20 Mbps	0.727 ms	21	208065
63	127	263.7 Mbps	20 Mbps	0.806 ms	3	207581

Table 6: Effect of contention window tuning on throughput.

UDP_{CW_MIN}	UDP_{CW_MAX}	TCP_{tx_cts}	TCP_{rx_cts}	UDP_{tx_rts}	UDP_{rx_cts}
15	31	40984	18831	88039	42582
31	63	45333	24371	48228	21167
63	127	45615	29033	23853	11821

Table 7: Counters of transmitted `RTS` from sources (`TCP`, `UDP`) and received `CTS` addressed to sources after contention window tuning.

As demonstrated by Table 6, contention window limit manipulation decreases number of non-replied RTS frames by UDP source. The expense of contention window increment is extra jitter in video traffic. Nevertheless, even video traffic has always been a lot important, less than 0.3 milliseconds extra jitter could be acceptable for not losing video packets, also for 130 Mbps improvement in best-effort traffic.

[8]Here we used Broadcom's `wl -i wl1 wme_ac ap vi ecwmin 15` (or ecwmax for maximum limit) command

Table 8: Modulation and coding schemes in 802.11ac

Index	Modulation	Coding Scheme	Rate (Mbps)
0	BPSK	1/2	6.5
1	BPSK	3/4	13.0
2	QPSK	1/2	19.5
3	QPSK	3/4	26.0
4	16 QAM	1/2	39.0
5	16 QAM	3/4	52.0
6	64 QAM	1/2	58.5
7	64 QAM	3/4	65.0
8	256 QAM	3/4	78.0
9	256 QAM	5/6	86.7

CHAPTER III

INVESTIGATIONS ON MULTI-USER MIMO FEATURE IN IEEE 802.11 AC NETWORKS

In this chapter, we present our studies on the MU-MIMO feature with a heterogeneous Wi-Fi network where 802.11ac and legacy 802.11n nodes coexist [10]. In MU-MIMO technique, the 802.11ac AP performs beamforming at the downlink in order to increase the total throughput to its clients. Before initiating a MU-MIMO transmission, the AP first sends an `NDP Announcement` frame including the estimated duration of the MIMO transmission [11, 12]. Then it performs channel sounding in order to be able to perform beamforming, and finally sends the data -Figure 23.

Here, we focused on following two concerns associated with MU-MIMO operation: The first one is the overhead of the channel sounding operation before MU-MIMO transmission, which increases linearly with the total number of clients.

The second one is with the coexisting legacy clients. Since the clients that are not recipients may be at the low gain directions of the formed beams, they may not sense

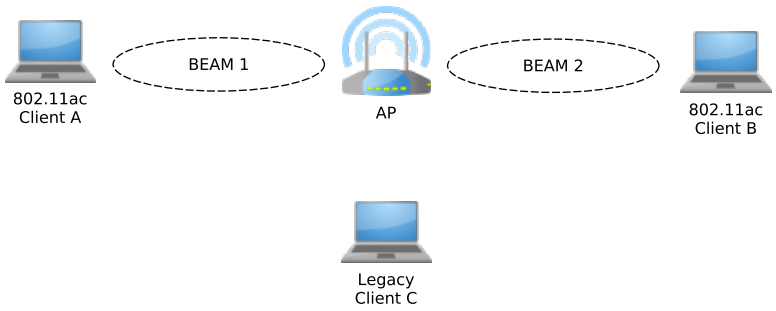

Figure 22: Illustration of deafness problem during MU-MIMO transmission.

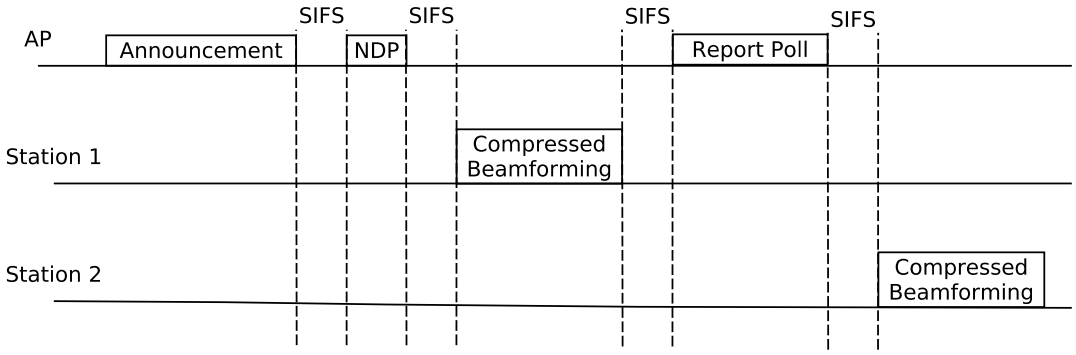

Figure 23: Channel Sounding Procedure before MU-MIMO transmission

the ongoing transmission, nevertheless, they have to remain silent for the announced duration of the MIMO transmission. However, since legacy clients are not able to decode announcement frame -backward compatibility problem- they can't be aware of the ongoing MU-MIMO transmission and initiate their own transmission, resulting in collision of both transmissions. This phenomenon is akin to the deafness problem in ad-hoc networks [13], see Figure 22.

In order to study the these problems, we first developed a simulator that models the PHY and MAC layers of 802.11ac and 802.11n networks in a detailed fashion. Below, we first describe construction of our simulation platform. Consequently, we present our studies on effect of the 802.11n users on the 802.11ac network during MU-MIMO transmission and overhead-information trade-off during channel sounding.

3.1 Simulation Platform

In order to analyze the overhead of channel sounding procedure, the effect of the deafness problem on the MU-MIMO transmissions and the proposed mitigation techniques, we developed a realistic, event-driven simulator that implements all aspects

of the PHY and MAC layers of 802.11n and 802.11ac protocols in a detailed fashion.

The channels between the nodes in the network (the AP and the clients) are modeled using the ITU indoor channel model [14]. The path loss in dB is computed according to

$$L_{\text{path}} = 20\log(f) + 10\alpha\log(d) + L_w(n) - 28, \tag{3}$$

where α is the path loss coefficient, f is the carrier frequency, d is the distance between nodes, and $L_w(n)$ is the loss due to concrete walls between the nodes as given in [14], where n is the number of concrete walls.

Log-normal shadowing with a standard deviation of 12 dB is assumed. A decorrelation distance of 10 meters and walking speeds of 0.33 m/s are assumed, which result in a shadowing coherence time of 30 s. The connections are assumed to be non-line-of-sight (NLoS), thus, Rayleigh Fading model is used. Considering walking speeds of 0.33 m/s and a carrier frequency of 5.2 GHz yield a Doppler frequency of 5.7 Hz, which results in a multi-path fading coherence time of 0.18 seconds. The beam patterns of the MU-MIMO transmission are formed according to the beam patterns for LTE antennas as given in [15].

The event driven simulation is carried out as follows. The data of the AP (downlink data) and the clients (uplink data) are assumed to arrive with packets that are uniformly distributed in time with a constant mean bitrate and enter the queues of the AP and the clients respectively. When the AP has data to send in its queue, it enters the CSMA procedure. After the back-off is complete, the AP either transmits data to one client or initiates a MU-MIMO transmission.

For this, if the channel parameters are not recent enough, the AP first performs channel sounding as explained above. Given the channel parameters, the AP decides on the optimal number of receiving clients and the beam pattern, including the option of sending to one client with no MIMO, that maximizes the throughput. At the end of

Table 9: PHY and MAC layer parameters used in simulations.

PHY LAYER PARAMETERS	
No. of antennas in AP, (M)	4
No. of antennas in client	1
No. of data sub-carriers in 802.11ac (N_{dsc})	468
Feedback period	10 ms
Channel model	ITU Office Area Model
Shadowing model	Log-normal
Shadowing standard deviation	12 dB
Shadowing coherence time	30 s
Multi-path fading model	Rayleigh
Multipath coherence time	0.18 s
Doppler frequency	5.7 Hz
Transmit power	17 dBm
Receiving sensitivity threshold	-82 dBm
Center frequency	5.2 GHz
Channel bandwidth (802.11ac)	160 MHz
Channel bandwidth (802.11n)	40 MHz

MAC LAYER PARAMETERS	
Aggregation scheme	A-MPDU
Transmission opportunity (TXOP) (802.11ac)	5500 μs
Transmission opportunity (TXOP) (802.11n)	2064 μs
Bit arrivals	Uniformly distr. in time
Average bit rate	Constant
Payload size (802.11ac)	12000 bits
Payload size (802.11n)	2000 bits
Retransmission limit	4
NDP frame duration	$36 + 4M\mu$s
NDP announcement frame	$152 + 16n$ bits
(n = no of clients involved in channel sounding).	
Compressed beamforming frame	$40 + (8MN_{\text{dsc}})$ bits
Beamforming report poll frame	96 bits
RTS frame	160 bits
CTS frame	112 bits
BAR frame	192 bits
BA frame	256 bits
ACK frame	112 bits
DIFS duration	34 μs
PIFS duration	25 μs
duration	16 μs
Slot duration	9 μs
Minimum contention window size	16 slots
Maximum contention window size	1024 slots

the MU-MIMO transmission, the AP transmits a `Block Acknowledgement Request` (BAR) frame to each client and clients respond with a `Block Acknowledgement` (BA) frame, according to which, the AP queue is updated.

The uplink process is also similar, where each client enters the CSMA procedure given data to be transmitted exists in its queue. After the back-off, the client either transmits the data directly or performs RTS/CTS handshake. Similar acknowledgement procedure follows data transmission. If a collision occurs, the collided frames indicated in the BA frame are kept for retransmission or dropped if retransmission limit is reached.

3.2 Investigation of Channel Sounding Overhead

As mentioned earlier, channel sounding performed by the IEEE 802.11ac AP for MU-MIMO beamforming has an overhead that increases linearly with the number of users. In our simulations, we first investigate this overhead. Toward this end, we propose two schemes with less overhead. In one scheme called `HALF`, instead of polling all clients, the AP polls at most half of the clients that have highest amount of data pending in the AP queue. In the second approach, the AP polls at most four clients in a similar fashion, which we call `FOUR`. The original scheme where all clients are polled is called `ALL`.

In all three schemes, after channel sounding, given the channel characteristics and the pending data, the AP decides on the best MU-MIMO beam pattern and other transmission parameters in order to maximize the total throughput achieved. This means, some clients that are along the same directions might not be simultaneously scheduled, or sometimes, MU-MIMO transmission may even be not optimal and the AP might fall back to scheduling a single user. Clearly, there is a trade-off here, which is the focus of this subsection. Polling more clients in the beginning incurs higher airtime overhead but it also increases the expected throughput by increasing

the possibility of scheduling more users with beamforming.

As a benchmark scheme, we also consider the case when no MU-MIMO transmission is performed and the AP schedules the client with the best channel characteristics. Although the channel information might not always be available without channel sounding, since this is a benchmark case to compare the other cases above, we idealistically assume that the overhead of this scheme is zero. We call this scheme `NONE`.

In this simulation, we assume that only downlink traffic with an average rate of 5 Mbps per client exists and we compare the performances of the four cases listed above. The network is assumed to be as illustrated in Figure 24 with 6, 12, 18 or 24 total clients distributed evenly around the circle.

We run the event-driven simulator to simulate a total duration of 10 seconds and the results are reported in Table 10, where, the columns report average downlink throughput, average number of MU-MIMO receivers, ratio of data packets that are transmitted with MU-MIMO transmission, average bonded channel bandwidth normalized to 20 MHz, and ratio of the time spent to channel sounding over the entire simulation duration, respectively.

Surprisingly, as the number of total clients increase, the benefit of MU-MIMO transmission is observed despite its increasing overhead. When there are a total of 6 clients, all methods satisfy the average traffic demand of 30 Mbps. However, as the number of clients increase toward 24, the throughput of performing sounding with all clients outperforms all other methods, although its channel sounding overhead becomes as high as 28%. This is because, due to having better channel information, a higher number of clients can be scheduled for transmission at a higher percent of the time, as observed from the average number of receivers (3^{rd}) and MU-MIMO ratio (4^{th}) columns. These benefits seem to pay-off the overhead of channel sounding, and for our simulation setting, when the number of clients in the network is large,

6 Clients	T. [Mbps]	Beam Average	MU-MIMO Ratio	BW. Ratio	Overhead Ratio
ALL	30.0	2.2	%52	7.44	%7
HALF	30.0	2.1	%34	7.51	%5
FOUR	30.0	2.1	%45	7.48	%6
NONE	30.0	-	-	7.49	-
12 Clients	T. [Mbps]	Beam Average	MU-MIMO Ratio	BW. Ratio	Overhead Ratio
ALL	59.0	2.5	%75	7.14	%21
HALF	53.8	2.3	%64	7.10	%10
FOUR	54.6	2.2	%41	7.23	%6
NONE	59.3	-	-	7.51	-
18 Clients	T. [Mbps]	Beam Average	MU-MIMO Ratio	BW. Ratio	Overhead Ratio
ALL	87.3	2.9	%97	7.38	%23
HALF	64.4	2.5	%86	7.34	%14
FOUR	57.0	2.2	%45	7.43	%7
NONE	72.0	-	-	7.72	-
24 Clients	T. [Mbps]	Beam Average	MU-MIMO Ratio	BW. Ratio	Overhead Ratio
ALL	111.2	3.2	%99	7.58	%28
HALF	72.0	2.7	%97	7.15	%17
FOUR	57.0	2.1	%47	7.39	%6
NONE	75.3	-	-	7.74	-

Table 10: Throughput Outcomes of downlink-only scenario

performing channel sounding with all the clients is always favorable.

3.3 Investigation of the Effect of Co-existing Legacy Clients

An IEEE 802.11ac AP performing MU-MIMO beamforming has an overhead that increases linearly with the number of users. In our simulations, we first investigate this overhead. To this end, we propose two schemes with less overhead. In one scheme called `HALF`, instead of polling all clients, the AP polls at most half of the clients that have highest amount of data pending in the AP queue. In the second approach, the AP polls at most four clients in a similar fashion, which we call `FOUR`. The original

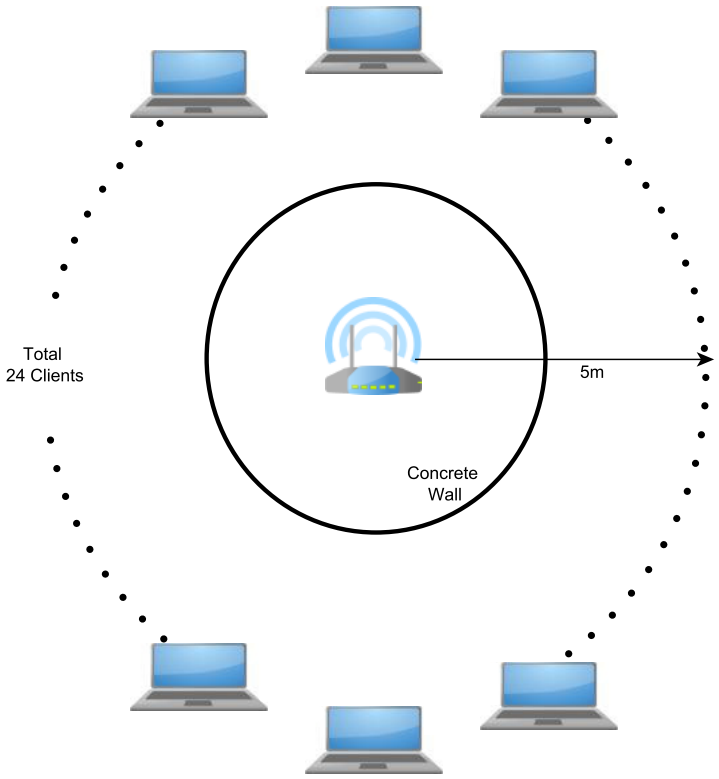

Figure 24: Graphical illustration of simulation constellation.

Table 11: Effect of deafness problem.

RTS/CTS usage	cts2self usage	Downlink [Mbps]	Uplink [Mbps]	Downlink Ratio	Uplink Ratio	Downlink Succ.	Uplink Succ.
✗	✗	11.1	11.5	%34	%66	%28	%84
✓	✗	45.5	8.3	%61	%38	%85	%58
✗	✓	45.0	11.9	%53	%47	%100	%100

scheme where all clients are polled is called `ALL`.

In all three schemes, after channel sounding, given the channel characteristics and the pending data, the AP decides on the best MU-MIMO beam pattern and other transmission parameters in order to maximize the total throughput achieved. This means, some clients that are along the same directions might not be simultaneously scheduled, or sometimes, MU-MIMO transmission may even be not optimal and the AP might fall back to scheduling a single user. Clearly, there is a trade-off here, which is the focus of this subsection. Polling more clients in the beginning incurs higher airtime overhead but it also increases the expected throughput by increasing the possibility of scheduling more users with beamforming.

As a benchmark scheme, we also consider the case when no MU-MIMO transmission is performed and the AP schedules the client with the best channel characteristics. Although the channel information might not always be available without channel sounding, since this is a benchmark case to compare the other cases above, we idealistically assume that the overhead of this scheme is zero. We call this scheme `NONE`.

In this simulation, we assume that only downlink traffic with an average rate of 5 Mbps per client is present and we compare the performances of the four cases listed above. The network is assumed to be as illustrated in Figure 24 with 6, 12, 18 or 24 total clients distributed evenly around the circle.

We run the event-driven simulator to simulate a total duration of 10 seconds and

the results are reported in Table 10, where, the columns report average downlink throughput, average number of MU-MIMO receivers, ratio of data packets that are transmitted with MU-MIMO transmission, average bonded channel bandwidth normalized to 20 MHz, and ratio of the time spent to channel sounding over the entire simulation duration, respectively.

The results of this experiment are given in Table 11. The first and second columns indicate whether the IEEE 802.11n clients use the regular RTS/CTS handshake, or the IEEE 802.11ac AP uses the `cts2self` mechanism. In the table, the third to last columns report the downlink and uplink throughputs, the ratio of downlink and uplink data transmission times over total transmission time, the ratio of successful downlink transmissions, and successful RTS frames or uplink transmissions of the legacy clients, respectively.

We observe that when neither RTS/CTS mechanism nor the `cts2self` mechanism is used, the IEEE 802.11ac downlink traffic is significantly impacted due to the deafness problem, as indicated by only 28% downlink success. When the legacy clients use the RTS/CTS mechanism, the 802.11ac traffic may still collide with the RTS frames, however, as RTS frames are short and actual uplink transmission does not start when RTS frames are unsuccessful, the IEEE 802.11ac downlink MU-MIMO traffic does not get effected from the deafness problem to a great extent. On the other hand, the IEEE 802.11n uplink performance drops due to consecutive RTS collisions. In the third row, we observe that usage of the `cts2self` mechanism by the AP provides better results in sum throughput, since no uplink and downlink frames collide as seen from the last two columns, and no airtime is wasted to collision.

CHAPTER IV

WI-FI PERFORMANCE INVESTIGATION WITH COEXISTENCE OF TCP

In this chapter, we investigated performance of TCP congestion control algorithms with coexistence of Wi-Fi when end-to-end connection consists lossy links. Before presenting our work, we first like to shortly review history and motivations of TCP - and its congestion control mechanism, from the days of a bit earlier than construction of first ever packet switched network.

4.1 The Story

In March 1950, several top scientists in geophysics (including Lloyd Berkner, Sydney Chapman, S. Fred Singer, and Harry Vestine) suggested that the time was ripe to have a worldwide Geophysical Year; especially considering recent advances -mostly from World War II- in rocketry, radar, and computing. Followingly, Berkner and Chapman proposed to the International Council of Scientific Unions that an International Geophysical Year (IGY) be planned for 1957-58, coinciding with an approaching period of maximum solar activity.

In 1955, President Eisenhower announced that, USA hoped to launch a small Earth orbiting satellite for gathering information about the upper atmosphere. The Kremlin announced that it hoped to do likewise. Planning in America focused on a sophisticated three stage rocket, but in Russia they took a more direct approach. Strapping four military rockets together, on 4 October 1957 the USSR launched Sputnik I (a 70 kgs bleeping sphere the size of a medicine ball) into Earth orbit.

The effect in the United States was electrifying, since this proved that Soviets had

rockets capable of sending nuclear weapons from Russia to Europe and even North America. One of the immediate reactions was the creation of the Advanced Research Projects Agency within the Ministry of Defence. Its mission was to apply state-of-the-art technology to US defence and to avoid being surprised (again!) by technological advances of the enemy. It was also given interim control of the US satellite program until the creation of NASA in October 1958.

ARPA became the technological think-tank of the American defence effort, employing directly a couple of hundred top scientists and with a budget sufficient for sub-contracting research to other top American institutions. Although the advanced computing would come to dominate its work, the initial focus of ARPA's activities were on space, ballistic missiles and nuclear test monitoring. Even so, from the start ARPA was interested in communicating between its operational base and its sub-contractors, preferably through direct links between its various computers[16].

4.2 Networking with its motivation

In 1962 ARPA opened a computer research program and appointed to its head a psychologist and computer scientist John Licklider from MIT to lead it. At that time, Licklider had an envision of "Galactic Network" concept to have more effective men&computer cooperation. The dream was having globally interconnected set of computers through which everyone could quickly access data and programs from any site. In May 1962, he pointed following tasks/requirements to do in "On-Line Man-Computer Communication" paper[17] that we extract from and provide below.

1. Develop systems for sharing the time of digital computers among many users on a split-millisecond basis.

2. Develop inexpensive electronic input-output surface which enables operator to communicate with computer.

3. Develop programming system to facilitate real-time contingent selection and shaping of information processing-procedures.

4. Develop systems for storage and retrieval of the vast quantities of information required to support, simultaneously at several user stations.

5. Solve the problem of human co-operation in the development of large program systems. It appears that the development of effective human cooperation and the development of man-computer symbiosis are "chicken-and-egg" problems. It will take unusual human teamwork to set up a truly workable man-computer partnership, and it will take man-computer partnerships to engender and facilitate the human cooperation.

Many efforts have been made to satisfy those requirements with respect to both political and industrial demands so far. For example, in the context of first requirement, development of Multics[1](Multiplexed Information and Computing Service) is trigged in DARPA funded Project MAC[2] (Multiple Access Computing), as time-sharing operating system. Below, we shortly introduce efforts to satisfy requirements 4 and 5 to stay in the scope of this thesis.

A major step along the path towards computer networking was from Leonard Kleinrock at MIT, since he published the first paper on packet switching theory in July 1961 and the first book on the subject in 1964. Consequently he convinced another MIT researcher Lawrence Roberts on theoretical feasibility of communications using packets rather than circuits. The other key step was to make the computers talk together. To explore this, Roberts and Merill connected the TX-2 computer in Massachusetts to the Q-32 in California with a low speed dial-up telephone line in 1965. This was the first (however small) wide-area computer network ever built. The

[1]Ancestor of Unix (Uniplexed Information and Computing System).
[2]Started at 1963

result of this experiment was the realization that the time-shared computers could work well together, running programs and retrieving data as necessary on the remote machine. Nevertheless, circuit switched telephone system was totally inadequate for the job. Kleinrock's conviction of the need for packet switching was confirmed[18].

4.3 Arpanet

In late 1966 Roberts went to DARPA and publish a plan for nationwide computer network system called "ARPANET", that will interconnect many dissimilar computers at ARPA-supported research centers with 50-kilobit common-carrier circuits. The primary goal of network is allowing persons and programs at one research center to access data and use programs that exist and run in other computers of the network. Below, we shortly provide basic working principles of Arpanet.

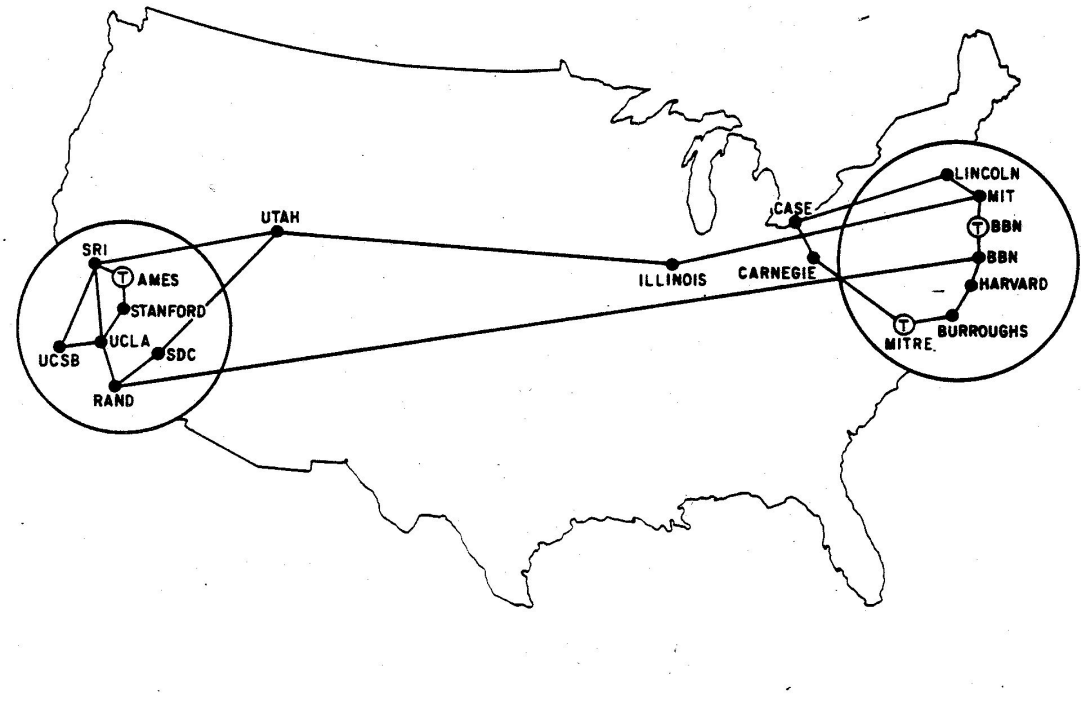

Figure 25: ARPANET (1971)

As illustrated in Figures 25 and 26, Arpanet was operating on non-fully connected

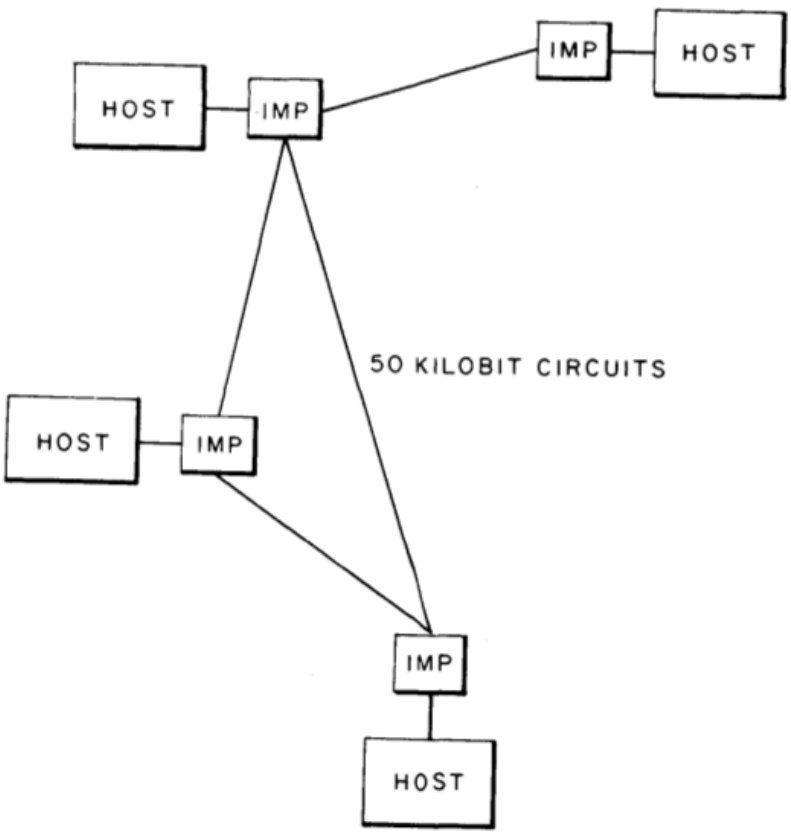

Figure 26: Interface Message Processors in Arpanet

network. The research computers[3] -that have time sharing operating system with dozens of consoles for multiple users- are connected to small processors. The subnet of this small processors (Interface Message Processor) stores copy of each message until next node has successful reception, handles routing, buffering, synchronization, error control and other related issues. The main reason of their existence was also insulating network related problems from research computers as explained in reference [19].

- Subnet should function as a *communications system* whose essential task is reliable bit transmission.

[3]Represented as Hosts in Figure 26

- Average transit time through the subnet should be under a half second to provide convenient interactive use of remote computers.
- Subnet should be completely autonomous. Since the subnet must function as a store and forward system, an IMP must not be dependent upon its local Host. The IMP must continue to operate whether the Host is functioning properly or not and must not depend upon a Host for buffer storage or other logical assistance such as program reloading.
- Establishment of Host-to-Host protocol and the enormous problem of planning to communicate between different computers should be an issue separated from the subnet design.

The working principle of Arpanet was a lot simpler than today's networks. The IMP subnet accepts only one message at a time on a given link[4]. So, source IMP does not introduce any further message towards same destination before getting acknowledgement from destination. After getting acknowledgement, he notifies Host by sending RFNM (*Ready for Next Message*) packet. By this simple design -also with maximum 8095 bit message size, Arpanet had preemptive mechanism to avoid congestion, which might be arised if destination Host would be flooded from multiple Hosts.

More details on routing, message handling etc. is available on reference [19].

4.4 Inter Network Communications

As we mentioned above, Arpanet became the basis for the Internet, which took advantage of the new idea of sending information in small units called *packets*. In order to expand the size of the network, Vinton Cerf and Robert[5] Kahn proposed a protocol that supports the sharing of resources that exist in different packet switching

[4]As it could be realized, this eliminates out of order problem.
[5]The duo said by many to be the Fathers of the Internet

networks. The protocol provides for variation in individual network packet sizes, transmission failures, sequencing, flow control, end-to-end error checking and the creation and destruction of logical process-to-process connections (via virtual ports); and called it as Transmission Control Protocol (TCP) [20]. Below, we briefly introduce the protocol.

4.4.1 TCP Overview

Initialization of TCP connection is handled by three negotiation messages, as follows:

1. Client sends a `SYN`chronize packet to Server
2. Server receives client's `SYN`
3. Server sends a `SYN`chronize-`ACK`nowledgement
4. Client receives server's `SYN-ACK`
5. Client sends `ACK`nowledge
6. Server receives `ACK`

After the sixth step, *TCP* connection state is called as *established*; and this procedure is called as three-way-handshake. At this stage, client breaks up messages into segments and transmits them to the server. Consequently, server reassembles the segments, even if they are received out of order.

Once client has "no more data to send", he sends a segment with the `FIN` bit set, to request that the connection be closed. The server receiving the `FIN` responds with an acknowledgement to the `FIN` to indicate that it was received. The connection as a whole is not considered terminated until both sides have finished the shut down procedure by sending a `FIN` and receiving an `ACK` -See Figure 27.

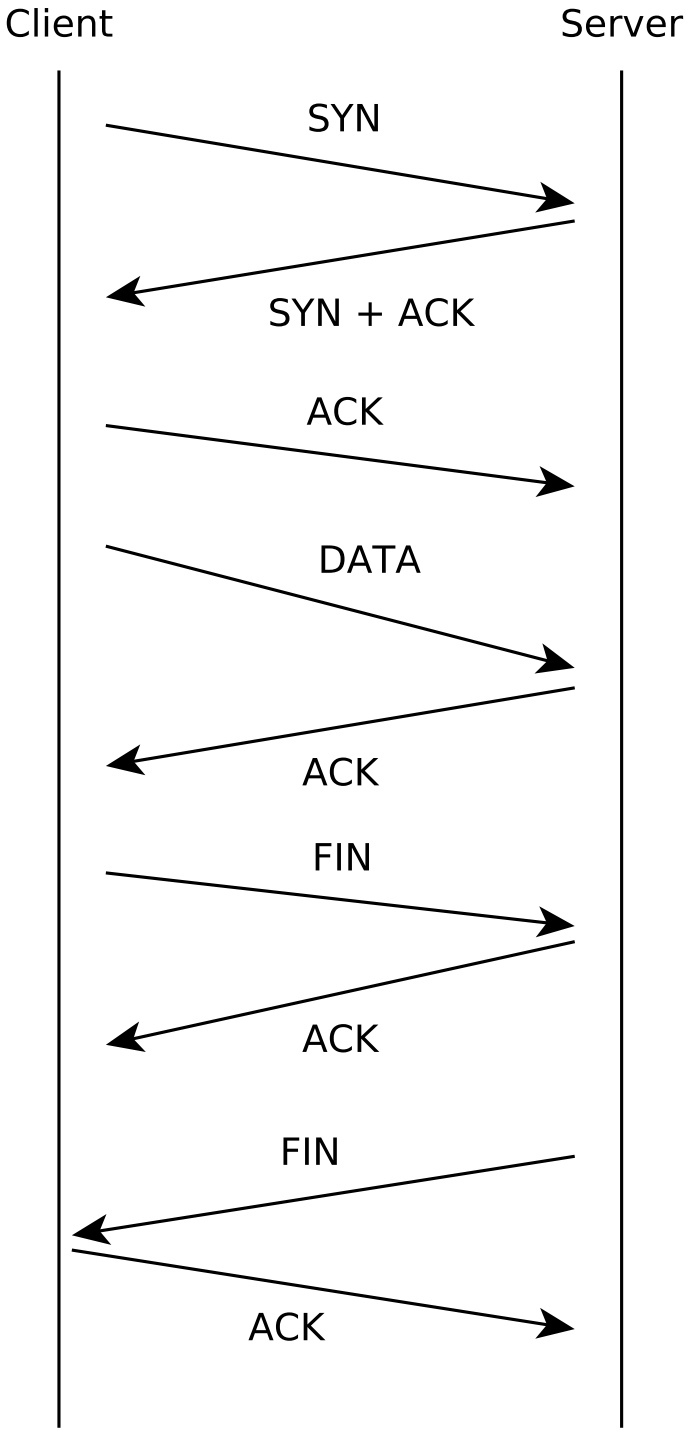

Figure 27: Packet Exchanges in Transmission Control Protocol

4.4.2 Congestion Control in TCP

As we mentioned above, clients send messages as segments and server reassembles them. Here, client would re-transmit the non-acknowledged data packets. Similarly, client might transmit multiple data packets without waiting an acknowledgement of each, then got acknowledgements -see Figure 28.

In case of having memory and/or processing bottleneck in any hop (usually routers) between end-to-end TCP connection, many packet drops and eventually re-transmissions would come up, which causes in-efficiency. For this reason, in 1988 Jacobson proposed a mechanism to avoid such circumstances in [21], which we call as *congestion control*.

The idea is tuning transmission number of un-acknowledged data packets, with respect to congestion detection. Here, the number of transmitted yet not acknowledged packets is called as *congestion window size*. In addition to this, *congestion detection* might be handled differently, with respect to delay and/or packet loss information. Combination of congestion-detection and window-size-manipulation is called as *congestion control algorithm*, which is handled in the kernel space of the operating system. For example, current algorithm in Linux could be seen via:

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```

Usually, congestion control algorithms additionally increases window size (w_i) upon acknowledgement reception -as in Equation 4, multiplicatively decreases window size upon congestion detection -as in Equation 5.

$$w_2 \leftarrow w_1 + a(w_1) \tag{4}$$

$$w_2 \leftarrow w_1(1 - b(w_1)) \tag{5}$$

As an example, we overview TCP Tahoe congestion control algorithm [22] below:

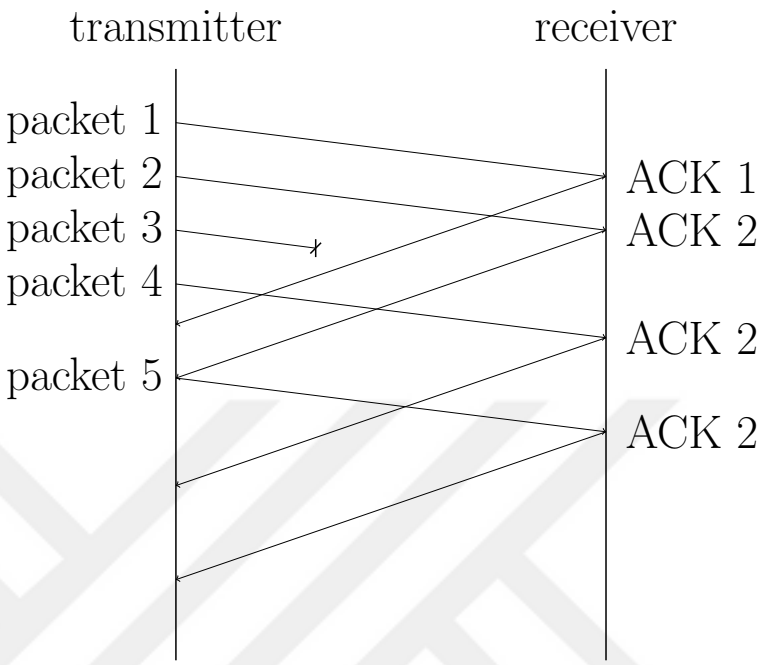

Figure 28: Illustration of Triple Duplicate ACK event.

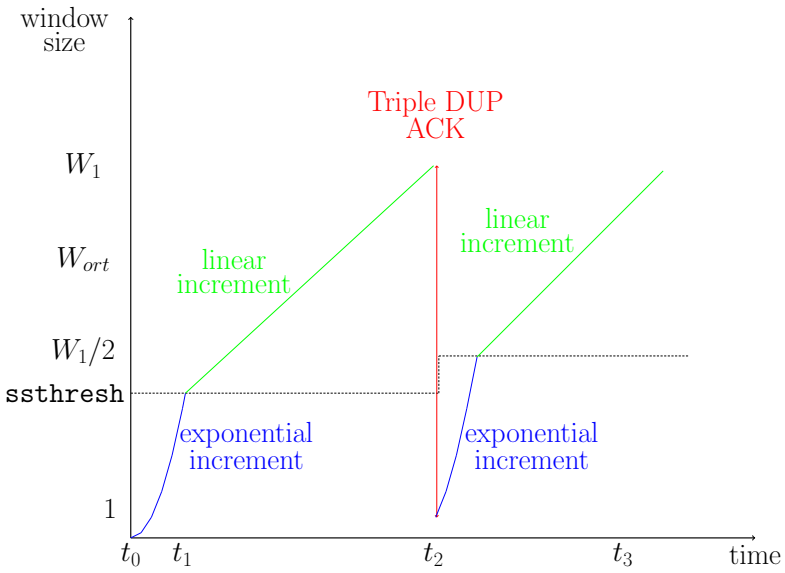

Figure 29: Graphical illustration of TCP Tahoe Congestion Control Algorithm.

- Until reaching the slow-start threshold, window size is exponentially (square function) incremented, upon acknowledgement reception.

- Once window size reaches the slow-start threshold, window size is linearly incremented one – $a(w_i) = 1$, upon acknowledgement reception.

- In case of lack of acknowledgement, transmitter would receive acknowledgement packets with same sequence numbers -since they increase cumulatively, See Figure 28. This fact is an indication of congestion, hence congestion window size would be halved -see Figure 29.

Below, we compare performance of said congestion-control algorithms in Wi-Fi network with existence of wireless interference.

4.5 Comparison of TCP Congestion Control Algorithms in Wi-Fi network

We considered performances of Scalable [23], HighSpeed [24], Cubic [25], Vegas [26], Veno [27], Yeah [28], Illinois [29] and Westwood [30] in Wi-Fi network with and without existence of interference [31].

We had the topology in Figure 30, and applied the following interference pattern in 100 second experiment:

- Transmitter 1 initiates a TCP traffic towards 802.11ac receiver.

- Transmitter 2 stays idle for initial 25 seconds.

- Transmitter 2 transmits UPD traffic towards 802.11n receiver for 5 seconds, then waits idle for 5 seconds; and repeats this procedure for ten times.

- Transmitter 2 stays idle for last 25 seconds.

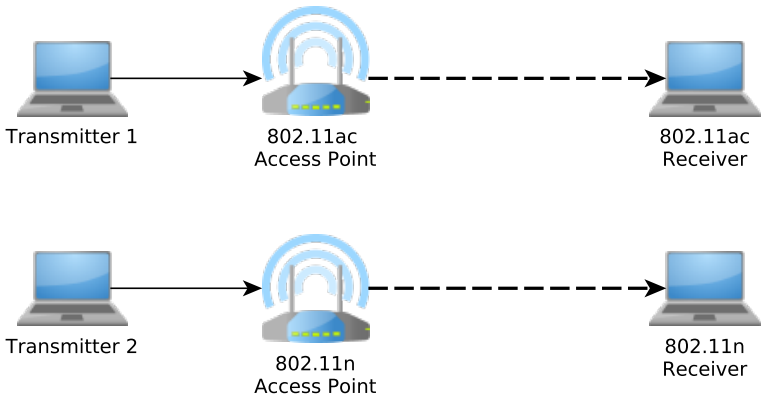

Figure 30: Representation of experiment topology.

Here, we considered the performance of aforementioned congestion control algorithms for 50 seconds, repeated the experiments without any interference and represent the output in Table 13. We used used loadable linux kernel module `tcpprobe` [32], to measure instantaneous congestion-window size statistics. Lastly, we also repeated the experiments with UDP traffic in 802.11ac networks to have a benchmark for the case.

Firstly, we observe that non of the TCP congestion control algorithms can reach the performance of UDP, even without any interference; and existence of interference -in our case- does not have a considerable impact on UDP traffic.

As we mentioned above, some congestion control algorithms use latency, some use packet loss, and some use both to detect congestion and decrease the window size. `Vegas` -highly depends on latency- had the worst output in our experiments, not able to saturate to link-capacity due to instantaneous latency differences even without any interference.

`Veno`, `Yeah`, `Illinois` and `Westwood` are the other congestion control algorithms that use latency as a parameter. We realized that `Illinois` and `Yeah` provide higher performance, due to their higher congestion window incrementation functions.

	Avg. Thr. [Mbps]	Max. Thr. [Mbps]	Min. Thr. [Mbps]	Std. Dev. Thr. [Mbps]
UDP	355.4	372	312	8.3
UDP*	355.2	374	330	7.4

Table 12: Throughput statistics of UDP protocol.

	Avg. Thr. [Mbps]	Max. Thr. [Mbps]	Min. Thr. [Mbps]	Std. Dev. Thr. [Mbps]	Avg. Window [Kbits]	Std. Dev. Window [Kbits]
Cubic	309.7	322.0	253.0	12.4	48.6	4.3
Cubic*	298.6	320.0	161.0	15.5	35.3	7.9
Vegas	49.2	55.6	37.7	3.9	0.33	0.06
Vegas*	42.7	58.7	25.2	11.7	0.32	0.06
H.Speed	305.9	316.0	255.0	10.0	21.1	2.1
H.Speed*	282.3	317.0	226.0	27.9	17.8	2.2
Illinois	312.6	321.0	278.0	7.9	22.8	3.3
Illinois*	295.9	318.0	245.0	20.4	21.9	2.9
Scalable	301.8	309.0	243.0	11.2	24.0	3.8
Scalable*	295.9	318.0	219.0	23.5	20.9	1.8
Veno	276.9	288.0	240.0	12.3	7.5	0.9
Veno*	261.8	302.0	177.0	37.5	7.5	0.9
Westwood	301.0	306.0	265.0	6.0	12.8	0.8
Westwood*	264.1	306.0	198.0	41.1	11.2	0.8
Yeah	297.0	304.0	267.0	8.3	10.2	1.2
Yeah*	272.1	306.0	201.0	41.0	10.2	1.1

Table 13: Comparison of throughput and congestion window size statistics in TCP congestion control algorithms with and without interference. The "*" notation is used to declare existence of interference.

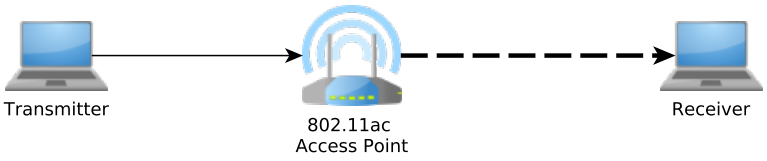

Figure 31: Representation of experiment topology.

`Scalable`, `Cubic` and `Highspeed` only use packet-loss as a parameter to detect congestion. After the detection of congestion, `Scalable` linearly increases window size, but `Highspeed` dynamically (concave or convex depending on internals of algorithm) increases the window size with respect to decreased window size. `Cubic`, also dynamically increases the window size, also uses the prior window-size before window-size degradation. We observed that, `Scalable` has better performance than `Highspeed`; `Cubic` has the best performance in the entire algorithm set.

4.6 Benefits of Multiple TCP connections

In this section, we considered performance of multiple TCP connections vs. single TCP connection with existence of virtually created packet loss scenarios. Here, we used `netem` (Network Emulation Tool) [33], that is supported by the Linux Foundation, to emulate loss with following commands:

- `tc qdisc change dev eth0 root netem delay 100ms 10ms 25%`
- `tc qdisc change dev eth0 root netem loss 2.5%`

In these experiments, we used `Cubic`, since, it's the default congestion control algorithm in linux kernel 2.6.19 and above. Our experiment topology is shown in Figure 31.

We observed that, multiple TCP connections provide nearly %10 improvement in case of not having packet-loss, as seen in Table 14. On the other hand, multiple TCP

# Connections	Avg. Thr. [Mbps]	Min. Thr. [Mbps]	Std. Dev. Thr. [Mbps]
1	300	212	33
2	327	252	19
3	322	244	27

Table 14: Delay: 0ms, Packet Loss Ratio: %0

# Connections	Avg. Thr. [Mbps]	Min. Thr. [Mbps]	Std. Dev. Thr. [Mbps]
1	143	35	36
2	241	49	55
3	283	128	58

Table 15: Delay: 0ms, Packet Loss Ratio: %2.5

# Connections	Avg. Thr. [Mbps]	Min. Thr. [Mbps]	Std. Dev. Thr. [Mbps]
1	88	12	31
2	170	42	39
3	237	106	46

Table 16: Delay: 0ms, Packet Loss Ratio: %5

connections provide significantly higher throughput, in case of having %2.5 and %5 packet-loss in TCP traffic, as shown in Tables 15 and 16.

The main reason for the such higher throughput is: Single TCP connection could not utilize the medium in case of having packet losses, due to multiplicative decrease, as we expressed above. Multiple connections, on the other hand, provides a higher possibility of higher congestion window sizes; since packet loss of one connection does not impact other's window size.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This thesis comprises performance improvement and degradation identification efforts in Wi-Fi networks. As it could be realized above, some problems has backward-compatibility origin and some has un-expected outcomes of particular features (such as aggregation and qos-prioritization in hidden-node scenario) in particular circumstances. Beside wireless communications & networking knowledge for intuition, all such work requires a lot experiment effort with patience, ability to find & use existing tools in open source environment and data mining.

Below, we provide today's "interesting" works in Wi-Fi networks, to improve user experience and network performance; with their challenges.

5.1 Wi-Fi Mesh Networks

This technology is not new at all, but became attractive after Google's interest for last two years. As it's very well known, number of mobile devices in home networks and video applications are daily increasing and seems to increase more due to diverse IoT devices. This requires more stable wireless connection in between clients and Wi-Fi network for better user experience –especially in homes built on stone. Most well known solution is multiple APs. The challenges and research activities in the field are:

- Routing: Considering mesh-backbone, such as existence of plc, ethernet, or multiple of them in between particular mesh nodes.
- Channel Selection: Considering dynamic frequency selection rules and mesh-backbone.

- Roaming: Both legacy clients that does not support fast roaming 802.11r and measurement 802.11k protocol.

5.2 Managing and Monitoring Wi-Fi Networks from Cloud

Understanding user-experience and diagnosing network from cloud is another interesting topic. One simple example would be, correlation of RSSI values and packet losses in Wi-Fi networks. Vendors and/or operators might offer next access point to the client with considering such data. Similarly, operators

5.3 Device Provisioning and Association

Adding third-party IoT devices to network, that might not have any input and/or limited state output -such as electric bulb- in a secure fashion. Device Provisioning Protocol task group of Wi-Fi alliance is working on standardization of such applications [34].

Bibliography

[1] J. Berg, "The ieee 802.11 standardization its history, specifications, implementations, and future,"

[2] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, "Ieee 802.11 ah: the wifi approach for m2m communications," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 144–152, 2014.

[3] B. Bellalta, "Ieee 802.11 ax: High-efficiency wlans," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 38–46, 2016.

[4] K. Çakmak, M. O. Sunay, A. Ö. Ercan, C. İlhan, C. Çakar, and Çağrı Sofuoğlu, "Experimental investigation of the impact on tcp traffic by hidden video-tagged udp traffic," in *Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2017 16th*, IEEE, 2017.

[5] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *htt p://dast. nlanr. net/Projects*, 2005.

[6] V. Jacobson, C. Leres, and S. McCanne, "The tcpdump manual page," *Lawrence Berkeley Laboratory, Berkeley, CA*, 1989.

[7] *pypcapfile, Python library for handling libpcap savefiles*, 2012.

[8] J. Berg, "Radiotap: De facto standard for 802.11 frame injection and reception."

[9] M. Bostock, "D3. js," *Data Driven Documents*, 2012.

[10] K. Çakmak, M. O. Sunay, and A. Ö. Ercan, "Investigations on multi-user mimo feature in ieee 802.11 ac networks," in *Signal Processing and Communication Application Conference (SIU), 2016 24th*, pp. 477–480, IEEE, 2016.

[11] M. Gast, *802.11ac: A survival guide*. O'Reilly Media, Inc., 2013.

[12] IEEE P802.11ac/D5.1, "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 4: enhancements for very high throughput for operation in bands below 6 GHz.," draft standard, IEEE, 2013.

[13] R. Ramathan, "Antenna beamforming and power control for ad-hoc networks," *Mobile Ad Hoc Networking S. Basagni et al. (eds)*, 2004.

[14] ITU-R, "Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 mhz to 100 ghz," technical report, ITU-12, 20012.

[15] B. Schulz, "LTE transmission modes and beamforming," technical report, Rohde & Schwarz, May 2014.

[16] R. T. Griffiths, "From ARPANET to World Wide Web." `http://www.let.leidenuniv.nl/history/ivh/chap2.html`, 2002. [Online; accessed 06-November-2016].

[17] J. C. R. Licklider and W. E. Clark, "On-line man-computer communication," in *Proceedings of the May 1-3, 1962, spring joint computer conference*, pp. 113–128, ACM, 1962.

[18] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.

[19] F. E. Heart, R. E. Kahn, S. Ornstein, W. Crowther, and D. C. Walden, "The interface message processor for the arpa computer network," in *Proceedings of the May 5-7, 1970, spring joint computer conference*, pp. 551–567, ACM, 1970.

[20] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 71–82, 2005.

[21] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM computer communication review*, vol. 18, pp. 314–329, ACM, 1988.

[22] J. F. Kurose, *Computer networking: A top-down approach featuring the internet, 3/E.* Pearson Education India, 2005.

[23] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," *ACM SIGCOMM computer communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[24] S. Floyd, "Highspeed tcp for large congestion windows," 2003.

[25] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[26] L. S. Brakmo and L. L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.

[27] C. P. Fu and S. C. Liew, "Tcp veno: Tcp enhancement for transmission over wireless access networks," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 2, pp. 216–228, 2003.

[28] A. Baiocchi, A. P. Castellani, and F. Vacirca, "Yeah-tcp: yet another highspeed tcp," in *Proc. PFLDnet*, vol. 7, pp. 37–42, 2007.

[29] S. Liu, T. Başar, and R. Srikant, "Tcp-illinois: A loss-and delay-based congestion control algorithm for high-speed networks," *Performance Evaluation*, vol. 65, no. 6, pp. 417–440, 2008.

[30] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "Tcp westwood: end-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 467–479, 2002.

[31] K. Çakmak, M. O. Sunay, A. Ö. Ercan, and M. Karaca, "Performance of tcp congestion control algorithms with interference in ieee 802.11 ac wlans," in *Signal Processing and Communication Application Conference (SIU), 2016 24th*, pp. 473–476, IEEE, 2016.

[32] S. Hemminger, "Tcp testing (tcpprobe)," *The Linux Foundation–http://devresources. linuxfoundation. org/shemminger/tcp, Accessed*, 2011.

[33] *netem, Network Emulation for testing protocols by emulating the properties of wide area networks.*

[34] WI-FI ALLIANCE, *Wi-Fi Device Provisioning Protocol Technical Specification*, 2016. Version 0.0.23.