NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TIAGO MALTIEIRA FERNANDES CORREIA

BSc in Electrical and Computer Engineering

DESIGN AND PERFORMANCE EVALUATION OF AN IEEE 802.11AX SYSTEM FOR CATHLAB WITH HIGH QOS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

DESIGN AND PERFORMANCE EVALUATION OF AN IEEE 802.11AX SYSTEM FOR CATHLAB WITH HIGH QOS

TIAGO MALTIEIRA FERNANDES CORREIA

BSc in Electrical and Computer Engineering

Adviser: Luís Filipe Lourenço Bernardo
Associate Professor, NOVA University Lisbon

Co-adviser: João Francisco Martinho Lêdo Guerreiro
Assistant Professor, NOVA University Lisbon

Examination Committee

Chair: Luís Filipe Figueira Brito Palma
Associate Professor, NOVA University Lisbon

Rapporteur: Paulo da Costa Luís da Fonseca Pinto
Associate Professor, NOVA University Lisbon

Adviser: Luís Filipe Lourenço Bernardo
Associate Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2023

Design and performance evaluation of an IEEE 802.11ax system for CathLab with high QoS

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.5) [17].

To the love of my life.

Acknowledgements

I want to thank my professor for all of his support throughout my dissertation and being patient with me. I also want to thank Paula Simão for helping me access the research building on the last day before it closed for summer vacation. Last but not least I want to express my gratitude towards my family and friends for all the support, and a special thank you to Mariana Santos. This work is funded by *Instituto de Telecomunicações* and *Fundação para a Ciência e Tecnologia* under the project UIDB/50008/2020.

Abstract

Wireless network communications are increasingly common in our society. The high number of devices leads to very dense scenarios with high interference, leading to a degradation in the Quality of Service (QoS) of their communication, which is extremely difficult to achieve with current communication standards. Wi-Fi 6 or IEEE 802.11ax was developed to overcome these difficulties, bringing new developments in wireless network communications, increasing transmission efficiency and improving throughput for these communications. With the increase in devices communicating over wireless networks, medical applications are being considered to make this transition, aiding doctors' day-to-day operations and facilitating cleaning. A Cath Lab currently needs to carry out examinations using video capture and transmission. The cable connecting the devices has to be sterilized and cleaned between patients. To simplify the use of these instruments, a transition to wireless communication is being studied. This dissertation analyzes the performance of a IEEE 802.11ax wireless network using the Network Simulator 3 version 3.34 (ns-3.34). It studies an interference scenario where the Cath Lab WiFi network connecting the devices coexists with a nearby interfering IEEE 802.11ax network, showing that the Cath Lab QoS is sustained as long as the interfering load is limited and a minimum distance to the Cath Lab AP is satisfied.

Keywords: IEEE 802.11ax, Wi-Fi, Network Simulator 3 (ns-3), Cath Lab, Spatial Reuse, Wireless Local Area Network (WLAN), Quality of Service (QoS),

Resumo

As comunicações de redes sem fios são cada vez mais comuns na nossa sociedade. O número elevado de dispositivos leva a cenários muito densos com elevada interferência, levando a uma degradação da Qualidade de Serviço (QoS) da comunicação dos mesmos, sendo extremamente difícil de alcançar com as normas de comunicação actuais. O Wi-Fi 6 ou IEEE 802.11ax foi desenvolvido para ultrapassar essas dificuldades, trazendo novos desenvolvimentos nas comunicações de redes sem fios, aumentando a eficiência de transmissão e melhorando o débito destas comunicações. Com o aumento de dispositivos a comunicar em redes sem fios, as aplicações médicas estão a ser consideradas para fazer essa transição, ajudando na operação corrente dos médicos e facilitando a sua limpeza. Um Cath Lab atualmente necessita de realizar exames com recurso a captura e transmissão de vídeo. O cabo que liga aos dispositivos tem que ser esterilizado e limpo entre pacientes. Para simplificar a utilização desses instrumentos é estudada uma transição para uma comunicação sem fios. Esta dissertação analisa o desempenho de uma rede sem fios com IEEE 802.11ax, utilizando o Network Simulator 3 versão 3.34 (ns-3.34). Estuda um cenário de interferência em que a rede WiFi do Cath Lab que liga os dispositivos coexistentes, com uma rede IEEE 802.11ax com interfererência nas proximidades, mostrando que a QoS do laboratório de catéter é mantida desde que a carga de interferência seja limitada e que seja respeitada uma distância mínima até ao AP do laboratório de catéter.

Palavras-chave: IEEE 802.11ax, Wi-Fi, Network Simulator 3 (ns-3), Cath Lab, Spatial Reuse, Wireless Local Area Network (WLAN), Quality of Service (QoS),

Contents

List of Figures

List of Tables

Glossary

ns-3 Network Simulator, a discrete-event network simulator for internet systems *(pp. 2, 18, 19, 25, 30, 46, 47)*

Acronyms

AP Access Point *(pp. 3, 5, 7–9, 11–13, 16, 17, 21, 22, 25, 26, 30, 35, 46)*
API Application Programming Interface *(p. 18)*
ARP Address Resolution Protocol *(pp. 25, 30)*

BA Block Acknowledgment *(p. 13)*
BAR Block ACK Request *(pp. 13, 22)*
BEB Binary Exponential Backoff *(p. 14)*
BPSK Binary Phase Shift Keying *(p. 10)*
BSS Basic Service Set *(pp. 3, 4, 15, 16, 20, 21, 25, 48)*

CBR Constant Bit Rate *(p. 25)*
CCA Clear Channel Assessment *(p. 12)*
CS Carrier Sensing *(p. 12)*
CSMA/CA CSMA with Collision Avoidance *(pp. 3, 8, 12, 14, 15)*
CSV Comma-Separated Values *(p. 26)*
CTS Clear To Send *(pp. 14, 15, 22)*
CW Contention Window *(p. 17)*

DCF Distributed Coordination Function *(pp. 3, 12, 14, 18)*
DCM Dual Carrier Modulation *(p. 10)*
DIFS Distributed Coordination Function Interframe Spacing *(pp. 12, 13)*
DL Downlink *(pp. 4, 8, 12, 13, 18)*

ED Energy detection *(p. 12)*
EDCA Enhanced Distributed Channel Access *(pp. 17, 18, 21)*
ER Extended Range *(p. 8)*

FFT Fast Fourier Transform *(p. 7)*

GI Guard Interval *(pp. 4, 6, 7, 11, 20, 22, 24, 25, 47)*

HCCA Hybrid Coordination Function Controlled Channel Access *(p. 17)*
HE High Efficiency *(pp. 1, 8, 9, 11, 12, 16)*

IFFT Inverse Fast Fourier Transform *(p. 5)*
IFS Interframe Spacing *(pp. 12, 17)*
IP Internet Protocol *(p. 30)*

LAN Local Area Network *(p. 3)*

MAC Medium Access Control *(pp. 1, 3, 5, 7, 12, 13, 18, 47)*
MCS Modulation and Coding Scheme *(pp. 10, 11, 19, 20, 22, 24, 25, 47, 48)*
MIMO Multiple-Input and Multiple-Output *(pp. 11, 22)*
MPDU MAC Protocol Data Unit *(pp. 18, 22)*
MSDU MAC Service Data Unit *(p. 18)*
MU Multi-user *(pp. 5, 9, 11, 13, 22)*
MU-MIMO Multi-user Multiple-Input Multiple-Output *(pp. 4, 9, 11, 12)*

NAV Network Allocation Vector *(pp. 4, 12, 15, 16)*

OBSS Overlapping Basic Service Set *(pp. 16, 19)*
OFDM Orthogonal Frequency-Division Multiplexing *(pp. 4–6)*
OFDMA Orthogonal Frequency-Division Multiple Access *(pp. 4, 5, 7–9, 12, 13, 17–19)*

PCF Point Coordination Function *(p. 12)*
PDU Protocol Data Unit *(p. 12)*
PE Packet Extension *(p. 9)*
PHY Physical Layer *(pp. 1, 3, 5, 12, 18, 47)*
PIFS Priority function Interframe Spacing *(pp. 12, 13)*
PPDU Physical Layer Protocol Data Unit *(pp. 8, 9, 12, 13)*
PSC-UL Proximity-based Sensitivity Control for UL *(p. 16)*
PSK Phase Shift Keying *(p. 10)*

QAM Quadrature Amplitude Modulation *(pp. 2, 10)*
QoS Quality of Service *(pp. 1, 2, 5, 17, 18)*

RAM Random Access Memory *(p. 20)*
RNG Random Number Generation *(p. 27)*
RTS Request To Send *(pp. 14, 15, 22)*

RU Resource unit *(pp. 7, 8, 12, 13)*

SCS Subcarrier Spacing *(p. 6)*
SIFS Short Interframe Spacing *(pp. 12, 13)*
SIG Signal Field *(pp. 8, 9, 11, 16)*
SNR Signal-to-noise ratio *(pp. 10, 19)*
SSID Service Set Identifier *(p. 3)*
STA Station *(pp. 3, 8, 9, 12, 13, 15–17, 21)*
STF Short Training Field *(p. 9)*
SU Single-User *(pp. 8, 9, 11, 12)*

TB Trigger-based *(pp. 9, 12)*
TF Trigger Frame *(pp. 9, 12, 13)*
TWT Target Wake Time *(pp. 4, 16)*
TXOP Transmission Opportunity *(pp. 8, 13, 16)*

UL Uplink *(pp. 4, 8, 9, 12, 13, 16–18)*
UORA Uplink OFDMA Random Access *(p. 16)*

WAP Wireless Access Point *(p. 3)*
WLAN Wireless Local Area Network *(pp. 1, 3–5, 16, 47)*
WNIC Wireless Network Interface Controller *(pp. 3, 18)*

1 Introduction

1.1 Motivation

Ever since the initial standardization of a technology that allowed wireless communication, it only took a few years for it to be widely commercialized and to grow to the Wi-Fi that is used today. Wireless communication is extremely advantageous for its flexibility, simplicity, mobility and accessibility. By removing the need of a cable and being able to stay connected while moving within a local area, has drastically improved the way people use communicating devices.

The amount of connected devices is estimated to grow from 3.9 billion in 2018 to 5.3 billion by 2023 [5]. Furthermore, the added services that require higher throughput such as Ultra High Definition for Audio-Video streaming are growing at an extremely high rate, this combined with an increase of connected devices and Access Points, leads to dense scenarios where Quality of Service (QoS) will be extremely difficult to achieve with current communication standards.

A CathLab is an examination room that uses machines equipped with medical imaging to perform diagnosis on patients. They usually require some sort of reliable communication between the various machines and to the doctor and even between the doctor and the patient through devices such as headphones. The need for a QoS in medical imaging is extremely important so that doctors can perform the best diagnosis.

The IEEE assigned the High Efficiency (HE) Wireless Local Area Network (WLAN) Task Group to design a new amendment for both the Physical Layer (PHY) and the Medium Access Control (MAC) sublayer for HE operation in frequency bands between 1GHz and 7.125GHz, for telecommunications and information exchange between systems in local and metropolitan area networks. The result was the new amendment the IEEE 802.11ax.

1.2 Objectives

The main goal for this dissertation study is to design and evaluate the performance of an IEEE 802.11ax system for CathLab with high QoS. The minimum requirements for this work are a constant bitrate of $400Mbps$, which theoretically can be achieved with the 802.11ax amendment, since the theoretical values of bit rate using 1024-Quadrature Amplitude Modulation (QAM) with a 5/6 code rate on a 160 MHz channel range from $1.17Gbps$, with a single spatial stream, up to $9.38Gbps$ with 8 spatial streams. To achieve this goal, specific models will be created and simulated using the ns-3 tool.

1.3 Contributions

The main contributions of this work centers around the achievable throughput of a WiFi 6 network in the presence of interfering networks, utilizing the ns-3 simulator, by analysing various parameters such as the physical medium, total simulation time of the experiments and how it impacts the obtained results, the distance between the nodes and the amount of interference in the interfering nodes. These simulations are described in Section 3.3.

All of the simulation results were obtained using the code developed in ns-3 and are available in Annex I.

1.4 Document Structure

This document contains other chapters: the State of the Art in 2, the Methodology and Results in 3 and the Conclusion in 4. The State of the Art's purpose is to inform the reader of technical aspects related to this thesis, starting with an introduction to the technology and going through technical details, as well as various aspects of QoS in the IEEE 802.11 standard. The Methodology presents and explains the simulation network used in this work. The Conclusion presents the generic conclusions of this work as well as a future work Section in 4.1.

2

State of the art

2.1 IEEE 802.11

The IEEE 802.11 is a WLAN standard that specifies a set of technologies for wireless communication protocols between MAC and PHY protocols. It uses radio waves to communicate at certain frequencies, most notably at the $2.4GHz$, $5GHz$ and $6GHz$. It consists in half-duplex communication which means communication is possible in both directions, only one direction at the same time. Most WLANs are deployed in infrastructure mode, which is the mode that will be discussed. In this mode, a Station (STA) communicates through a Wireless Access Point (WAP) such as a router that serves as a bridge to usually connect to the Internet, in case of a private network to a Local Area Network (LAN). Multiple WAPs can be connected to expand the range of the network with or without the need of a wire.
Any component that connects to the network needs a Wireless Network Interface Controller (WNIC). The WAP or Access Point (AP) for short, creates an Infrastructure Basic Service Set (BSS) for other STAs to access. BSS is visible in Figure 2.1. It is a group of devices that are connected and share the same Service Set Identifier (SSID). They share the PHY characteristics to access the medium, such as radio frequency, modulation scheme and even security settings.
For a MAC mechanism, the IEEE 802.11 standard defines the Distributed Coordination Function (DCF) which uses the CSMA with Collision Avoidance (CSMA/CA) protocol [7]. This protocol is explained in further detail in Section 2.3.1 and tries to avoid collisions which are detrimental for wireless communication.
As the initial standard evolved throughout the years, many amendments were introduced and currently the most recent one is the IEEE 802.11ax amendment.

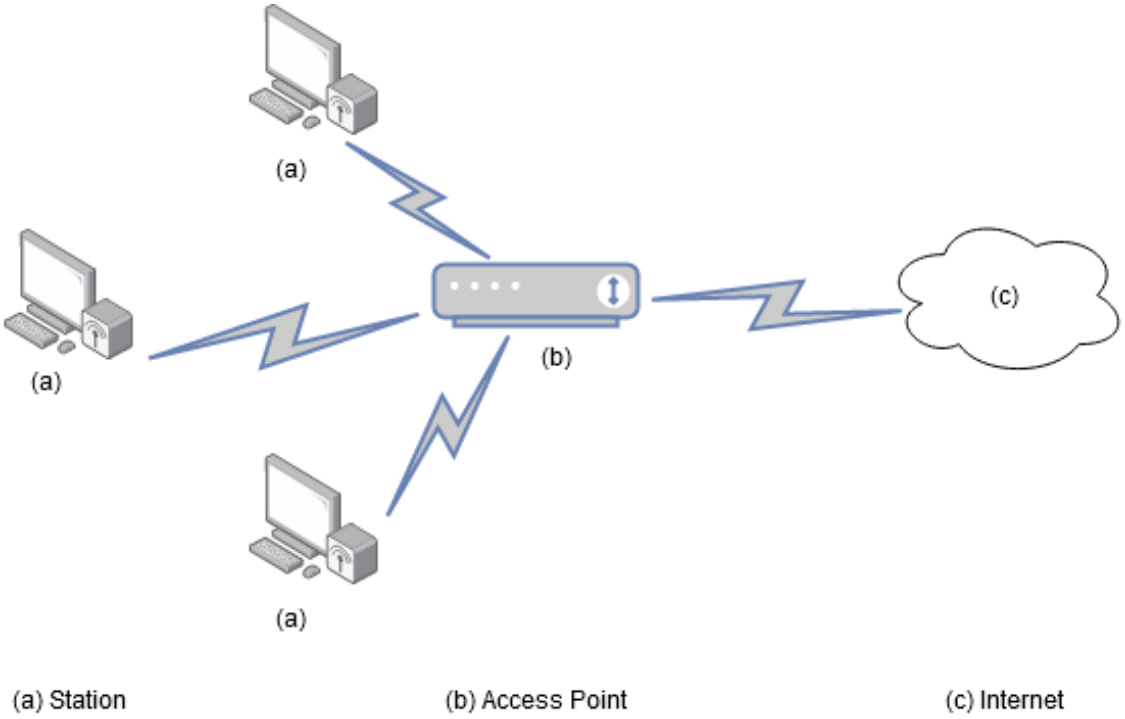


Figure 2.1: Basic Service Set

2.1.1 802.11ax

The IEEE 802.11ax amendment which is commonly referred to as Wi-Fi 6, improved and carried on the torch of previous WLANs amendments such as 802.11n and 802.11ac. This is an incremental improvement since these amendments introduced new features that the new 802.11ax amendment further improves, such as Orthogonal Frequency-Division Multiple Access (OFDMA) which extends Orthogonal Frequency-Division Multiplexing (OFDM) that was already used by both previous standards. While the 802.11ac amendment introduced Multi-user Multiple-Input Multiple-Output (MU-MIMO) for Downlink (DL), the 802.11ax enables Uplink (UL) MU-MIMO as well as DL. The Network Allocation Vector (NAV) has some improvements, namely the addition of dual NAV which ensures accurate virtual carrier sense and reduces collisions and thus increases spectral efficiency. The fragmentation of data packets also changes from a static fragmentation to a dynamic fragmentation, Guard Interval (GI) duration increased as well as symbol duration which is quadrupled. But the biggest innovation is in the implementation of new features such as BSS Coloring and Target Wake Time (TWT). The former allows to identify a BSS with a number, so when a frame from a different BSS is detected it is discarded. This is especially useful in dense scenarios, while the latter was first developed with the 802.11ah amendment, and can achieve lower power consumption by increasing device sleep time as well as optimize the spectrum efficiency. Every feature mentioned above is explained in further detail in a section below. We can observe in the Table 2.1 the main features and a comparison between the 802.11ax amendment and the previous ones (802.11ac and

802.11n).

Table 2.1: Comparison of 802.11ax with previous amendments

	802.11n	802.11ac	802.11ax
Frequency Bands	2.4GHz and 5GHz	5GHz	2.4GHz and 5GHz
Wave Form	OFDM	OFDM	OFDMA
Modulation	BPSK, QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM, 256-QAM	BPSK, QPSK, 16-QAM, 64-QAM, 256-QAM, 1024-QAM
OFDM Symbol time	3.2μs	3.2μs	12.8μs
Channel Width	20/40 MHz	20/40/80/160 MHz	20/40/80/160 MHz
Subcarrier spacing	312.5KHz	312.5KHz	78.125KHz
Guard Interval	0.4 or 0.8μs	0.4 or 0.8μs	0.8, 1.6 or 3.2μs
Total symbol time	3.6 or 4.0μs	3.6 or 4.0μs	13.6, 14.4 or 16.0μs
MIMO Order	4	8	8
MU-MIMO	N/A	Downlink only	Downlink and Uplink
Spatial Streams	4	8	8
Number of NAVs	1	1	2

The 802.11ax amendment faces mainly two new challenges. It is able to address dense scenarios since there is an increase on the deployment of new APs everywhere. And it is also able to handle the ever-increasing throughput needs for the upcoming years. To battle these challenges the 802.11ax amendment developed new PHY and MAC layer enhancements to continue the improvement of the QoS in WLANs. [3]

2.2 Physical Layer

The PHY in the 802.11ax amendment introduces improvements to OFDM, which is used in 802.11n and 802.11ac. The 802.11ax amendment improves it by introducing OFDMA, which is the Multi-user (MU) version of OFDM. To understand OFDMA there is a need to have some knowledge on how the previous OFDM works.

2.2.1 OFDM

OFDM is a modulation scheme for multiplexing signals to send them through the medium. It starts by dividing the channel bandwidth in multiple subcarriers that forward data independently from each other through an Inverse Fast Fourier Transform (IFFT) [29]. These subcarriers are really close to each other in the frequency domain but can still maintain their orthogonality. They are so close that each signal from each subcarrier extends onto each other as visible in Figure 2.2, although each subcarrier is designed so that on the peak of one OFDM subcarrier the others are equal to zero. This method

provides an efficient usage of the available bandwidth, since it allows a larger number of subcarriers for the same bandwidth.

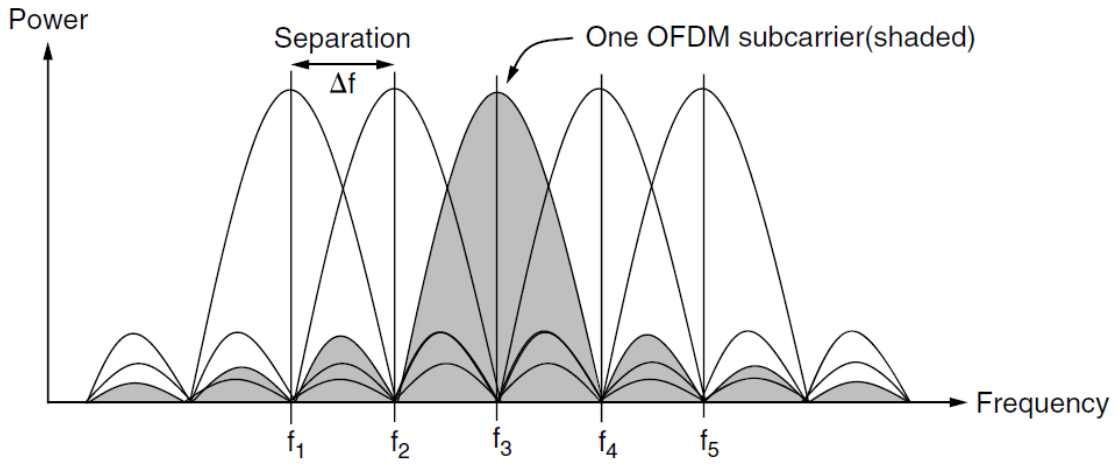


Figure 2.2: Orthogonal Frequency division multiplexing (OFDM)
Figure retrieved from [31]

With this scheme, it is possible to sample the subcarriers at their center or peak frequencies and have no interference from the other subcarriers. A GI is necessary to repeat a portion of the signal to achieve the desired frequency response. The space between subcarriers is orthogonal, so it uses carrier waves with different frequency to place the data next to each other in the available frequency space in the transmission medium. This orthogonality is obtained using Subcarrier Spacing (SCS):

$$\Delta f = \frac{k}{T_{SYM}}, \tag{2.1}$$

where T_{SYM} is the useful symbol duration in seconds and k is a positive integer usually equal to 1.

The division of the channel is extremely valuable, since some subcarriers can have degradation and may be excluded, while allowing other subcarriers to perform well. Furthermore, OFDM utilizes the available spectrum more efficiently and resists multipath propagation which leads to signal degradation. The frames are sent over multiple subcarriers in parallel, where most are used for carrying data and the rest are for synchronization. In a 20MHz channel it is possible to have 64 subcarriers by spacing them 312.5kHz, 52 are data subcarriers, 4 are pilot subcarriers and 8 unused subcarriers. With OFDMA there is a narrower spacing between subcarriers of only 78.125kHz, allowing 256 subcarriers. The longer symbol duration introduced with 802.11ax allows for this smaller subcarrier spacing and size causing an enhanced robustness in the channel and a better equalization. This is due to the longer symbol duration which decreases the subcarrier spacing and size, which allows for an enhanced robustness in the channel and a better equalization.

There are 4 different subcarrier types in an OFDM symbol, not all of them transport useful data. They are:

- The data subcarrier which is the actual data transmission;
- The pilot subcarrier that is used for synchronisation and channel estimation;
- The null subcarrier, this subcarrier has no data transmission and this frequency is used as a guard band;
- Direct Current subcarrier, the frequency of this subcarrier is equal to the centre frequency of the radio that is transmitting. Its frequency is equal to zero if the Fast Fourier Transform (FFT) signal is not modulated.

In 802.11ax, OFDMA operates using 20, 40, 80 and 160 MHz channel widths and can also use two non-adjacent 80 MHz channels. The duration of the symbols has quadrupled from 3.2 μs to 12.8 μs, therefore the symbols are less prone to jitter. Furthermore, with the increase in symbol duration there is also a decrease in overhead compared to the previous amendments. The GI which in previous amendments features a duration of 0.4 or 0.8 μs, in the 802.11ax amendment the duration is increased to 0.8 or 1.6 or even 3.2 μs which can reduce the overhead to 6% from 12-25% of the previous 802.11ac standard [13]. With the simultaneous transmission there is a decrease in overhead at the MAC sublayer. By using OFDMA we can achieve better use of the frequency and reduce latency which increases efficiency.

Table 2.2: Subcarriers and channels for OFDMA 802.11ax.
Adapted from [24]

Channel (MHz)	Sub-Channels	Largest Bandwidth (MHz)	Data Subcarriers w/o pilot	Lower-end Guard Subcarriers	Higher-end Guard Subcarriers	DC Subcarriers	Pilot Subcarriers	Null Subcarriers	Total Subcarriers
20	9*26	1.88	216	6	5	7	18	4	256
	4*52, 1*26	3.91	216	6	5	7	18	4	
	2*106, 1*26	7.97	228	6	5	7	10	0	
	1*242	18.3	234	6	5	3	8	0	
40	18*26	1.88	432	12	11	5	36	16	512
	8*52, 2*26	3.91	432	12	11	5	36	16	
	4*106, 2*26	7.97	456	12	11	5	20	8	
	2*242	18.3	468	12	11	5	16	0	
	1*484	36.6	468	12	11	5	16	0	
80	37*26	1.88	888	12	11	7	74	32	1024
	16*52, 5*26	3.91	888	12	11	7	74	32	
	8*106, 5*26	7.97	936	12	11	7	42	16	
	4*242, 1*26	18.3	960	12	11	7	34	0	
	2*484, 1*26	36.6	960	12	11	7	34	0	
	1*996	76.6	980	12	11	5	16	0	

With OFDMA it is possible to divide the channel in even smaller subchannels as it is visible in Table 2.2, so that low data-rate traffic can reduce its contention and header overhead. This also allows for an AP to bundle them in what is called a Resource unit (RU).

With the CSMA/CA to access the channel for each Transmission Opportunity (TXOP), there is wasted time on the medium, which in turn reduces efficiency. By grouping transmissions there is less time wasted for negotiation which increases efficiency. This is achieved because OFDMA assures contention free transmission to multiple STAs with one single TXOP for both DL as well as UL. In DL OFDMA it is possible with a single DL transmission from the AP to multiple STAs by splitting the frequency within a channel.

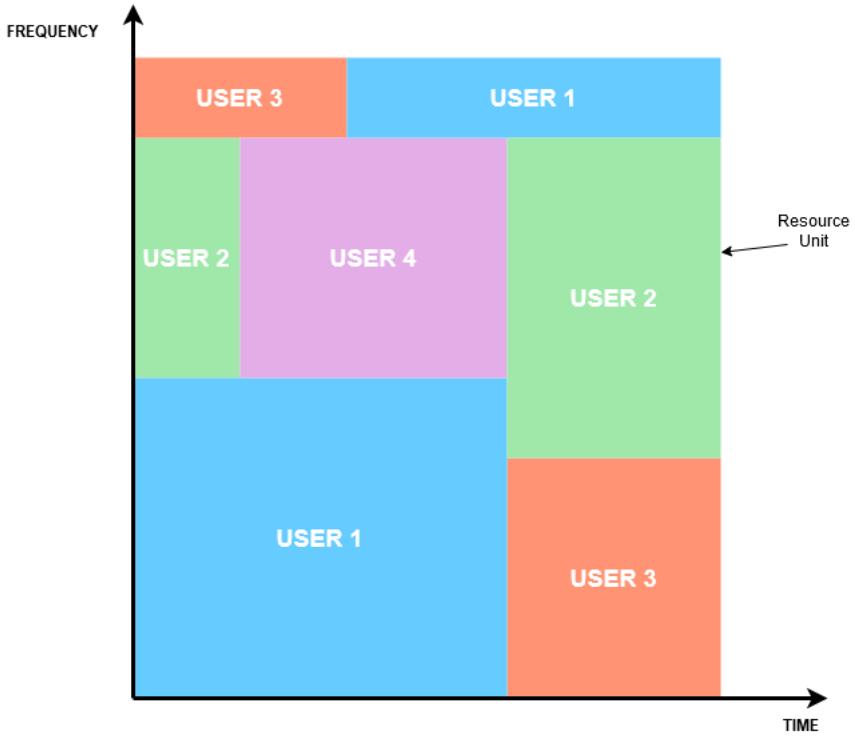


Figure 2.3: Orthogonal Frequency Division Multiple Access (OFDMA)

2.2.2 Physical Layer Protocol Data Unit

The 802.11ax standard defines 4 different Physical Layer Protocol Data Unit (PPDU) formats:

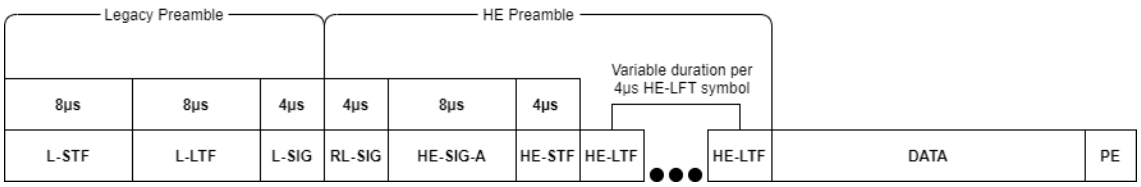
1. The HE Single-User (SU) PPDU, which is designed for SU transmission, i.e. to a single AP or STA;

2. The HE Extended Range (ER) SU PPDU, which is similar to the previous format, is used for longer range communication. This PPDU format differentiates from the others by having a longer HE-Signal Field (SIG)-A field that repeats each symbol while also having a power boost in the preamble for longer coverage, hence the name. This format is limited to only one spatial stream and can only use the 106 or 242 tone RU on the primary 20MHz channel;

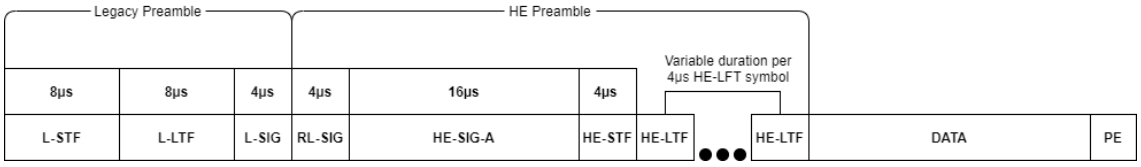
3. The HE MU PPDU is designed for MU transmission and thus for MU-MIMO/OFDMA. This format is the only that uses the HE-SIG-B field to assign multiple STAs in one data unit. A STA can also transmit to the AP if it supports this reception;

4. Finally the HE Trigger-based (TB) PPDU is used in response to a Trigger Frame (TF) for a UL MU transmission. Like the HE SU PPDU they are almost equal, barring the longer HE Short Training Field (STF) field.

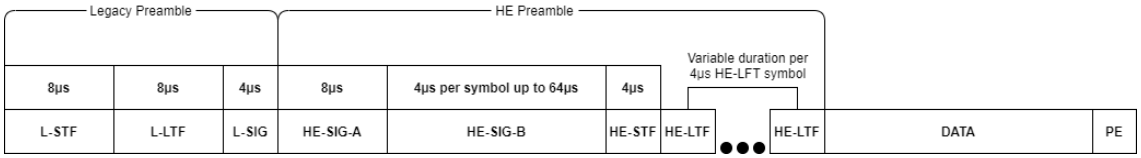
They are illustrated in the following Figure 2.4.



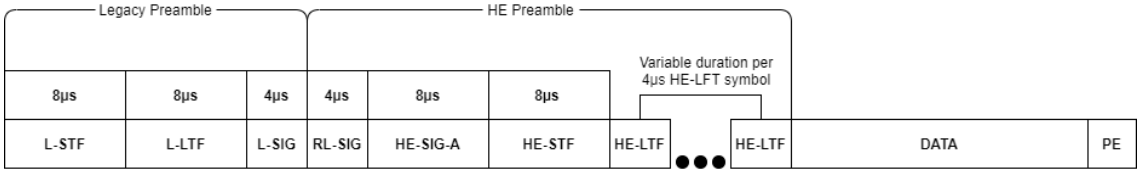
(a) HE SU PPDU



(b) HE ER SU PPDU



(c) HE MU PPDU



(d) HE TB PPDU

Figure 2.4: PPDU Formats adapted from [32]

All 4 PPDU formats can carry a Packet Extension (PE) field attached to the end. This field which can have durations of 0, 4, 8, 12 or 16 μs, is used so the receiver has enough time to obtain the packet. The PE has the same average transmit power of the data field, its content serves only to occupy the air interface time and so its content is arbitrary when used [9].

The legacy portion of the preamble's purpose is to include backwards compatibility and coexistence with previous amendments. The HE preamble can only be decoded by 802.11ax devices.

2.2.3 Modulation and Coding Scheme

By modulating the signal, it is possible to increase the amount of bits sent with each symbol. In digital modulation it is possible to modulate frequency, phase and amplitude. The simplest form of Phase Shift Keying (PSK) is Binary Phase Shift Keying (BPSK) which uses two phases for each symbol period. As modulation schemes evolved, QAM was introduced which modulates both phase and amplitude and can achieve higher spectrum efficiency. A QAM signal carries two waves with a phase difference of 90°, hence the name quadrature, that are modulated in amplitude and combined with phase variation, therefore increasing the effective bandwidth. A basic signal can only have two positions, 0 or 1. With 1024-QAM it is possible to achieve a constellation diagram like in Figure 2.5 by defining values of phase and amplitude.

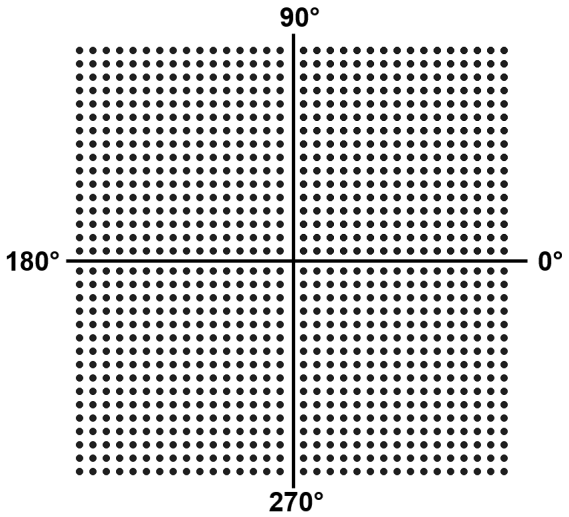


Figure 2.5: Constellation Diagram of 1024-QAM

With 1024-QAM which was introduced in the 802.11ax amendment, it is possible to modulate 10 bits per symbol as opposed to 256-QAM introduced in the 802.11ac amendment that modulates 8 bits per symbol. This is a potential 25% increase in throughput, although this comes at the cost of needing higher Signal-to-noise ratio (SNR) thresholds. The constellation points, which are done with gray bit mapping are more susceptible to noise, so for longer ranges, 256-QAM can be more reliable. There are currently 11 Modulation and Coding Scheme (MCS) which are listed in the Table 2.3. Dual Carrier Modulation (DCM) is introduced with the 802.11ax amendment and is used to enhance performance by enhancing robustness and protection against errors in regions with a low SNR [4].

MCS	Modulation	Dual Carrier Modulation	Coding Rate
0	BPSK	X	1/4
0	BPSK		1/2
1	QPSK	X	1/4
1	QPSK		1/2
2	QPSK		3/4
3	16-QAM	X	1/4
3	16-QAM		1/2
4	16-QAM	X	3/8
4	16-QAM		3/4
5	64-QAM		2/3
6	64-QAM		3/4
7	64-QAM		5/6
8	256-QAM		3/4
9	256-QAM		5/6
10	1024-QAM		3/4
11	1024-QAM		5/6

Table 2.3: Modulation Coding Scheme

The IEEE 802.11ax amendment introduces new MCS to enhance spectral efficiency The MCS index is transported in the HE-SIG-A field for SU transmission and for MU transmission in the HE-SIG-B field.

In Table 2.4 adapted from [19] are the theoretical values for achievable throughput across all MCS values as well as Channel Width and GI. These values are for only one spatial stream. Since 802.11ax allows up to 8 spatial streams the theoretical values can increase by eightfold.

Table 2.4: Theoretical Throughput across Modulation Coding Scheme in Mbps

	20 MHz			40 MHz			80 MHz			160 MHz		
MCS Index	3.2μs GI	1.6μs GI	0.8μs GI	3.2μs GI	1.6μs GI	0.8μs GI	3.2μs GI	1.6μs GI	0.8μs GI	3.2μs GI	1.6μs GI	0.8μs GI
0	7.3	8.1	8.6	14.6	16.3	17.2	30.6	34	36	61.3	68.1	72.1
1	14.6	16.3	17.2	29.3	32.5	34.4	61.3	68.1	72.1	122.5	136.1	144.1
2	21.9	24.4	25.8	43.9	48.8	51.6	91.9	102.1	108.1	183.8	204.2	216.2
3	29.3	32.5	34.4	58.5	65	68.8	122.5	136.1	144.1	245	272.2	288.2
4	43.9	48.8	51.6	87.8	97.5	103.2	183.8	204.2	216.2	367.5	408.3	432.4
5	58.5	65	68.8	117	130	137.6	245	272.2	288.2	490	544.4	576.5
6	65.8	73.1	77.4	131.6	146.3	154.9	275.6	306.3	324.3	551.3	612.5	648.5
7	73.1	81.3	86	146.3	162.5	172.1	306.3	340.3	360.3	612.5	680.6	720.6
8	87.8	97.5	103.2	175.5	195	206.5	367.5	408.3	432.4	735	816.7	864.7
9	97.5	108.3	114.7	195	216.7	229.4	408.3	453.7	480.4	816.7	907.4	960.8
10	109.7	121.9	129	219.4	243.8	258.1	459.4	510.4	540.4	918.8	1020.8	1080.9
11	121.9	135.4	143.4	243.8	270.8	286.8	510.4	567.1	600.5	1020.8	1134.3	1201

2.2.4 MU-MIMO

Multiple-Input and Multiple-Output (MIMO) technology utilizes beamforming to support various streams of data from one AP to multiple users [29]. MU-MIMO which is the MU version of MIMO, was first introduced in the IEEE 802.11ac amendment and its goal was to improve total network throughput, which it did. MU-MIMO along with static and dynamic

channel bonding allowed this amendment to achieve transmission rates in the order of gigabits, although as mentioned previously it was only available as DL [2]. It was the first amendment that allowed for up to four STAs to work in DL simultaneously. It is defined in [12] as "a technique where multiple STAs each with potentially multiple antennas can transmit and/or receive independent data streams simultaneously". It chooses orthogonal channels and concurrent transmissions to not cause interference between users. The 802.11ax amendment introduces MU-MIMO with UL capabilities, and in [26] concluded that for SU transmissions, MU-MIMO with UL OFDMA can improve the throughput by 474% as compared to just UL OFDMA which only improves the throughput by 273% as compared to the previous 802.11ac amendment throughput. With DL MU-MIMO it is possible for an AP to use 8 spatial streams and serve at most 4 STAs for every RU available. The AP begins by transmiting a Null Data Packet Announcement, subsequently the Null Data Packet and finally a Beamforming Report Poll [30]. For a UL MU-MIMO transmission, the AP starts the transmission with a TF and the STAs respond with their data using the HE TB PPDU as described in Section 2.2.2. For this transmission, the RU size has to be 106 or more tones, moreover UL MU-MIMO can only be used with scheduled access. If a RU has two or more STAs transmitting simultaneously, on the same random access, the data will be lost [8].

2.3 MAC Sublayer

The MAC sublayer is responsible for allocating the channel, for addressing the Protocol Data Unit (PDU), checking errors and formatting frames which includes fragmentation and reassembly of packets. The IEEE 802.11 standard defines two MAC sublayers: The Point Coordination Function (PCF) which was optional and is now obsolete, will not be discussed here.

The DCF uses CSMA/CA to avoid collisions in 802.11 communication. As the name suggests the protocol senses the availability of the channel using Clear Channel Assessment (CCA) at the PHY before transmitting any information, in order to avoid collisions.
This assessment can be made by detecting the energy level and/or by Carrier Sensing (CS). In the case of Energy detection (ED), the medium is measured by the total energy received by a station ignoring whether the symbol is a valid 802.11 symbol. CS can be done either at the physical level, by measuring the strength of the received symbol from a valid 802.11 symbol, or it can be done at the virtual level via the NAV mechanism. The NAV contains information about the medium such as idle/busy and in case of busy it also has the amount of time remaining until the end of transmission. Both the PHY CS and the ED mechanisms have a different threshold for the amount of energy to consider the medium as busy. The standard MAC protocol of the IEEE 802.11 standard defines Interframe Spacing (IFS) as the period of time that a channel must remain idle so that a station can begin its "transmission". There are three basic interframe spacings such as: Short Interframe Spacing (SIFS), Priority function Interframe Spacing (PIFS) and Distributed Coordination

Function Interframe Spacing (DIFS).

- **SIFS (Short Interframe Space)** - It is used to separate transmissions that belong to a single dialogue and corresponds to the smallest Inter-Frame Space. This value is fixed and is calculated in such a way that an emitting station returns to the reception mode and is able to decode a packet to receive it.

- **PIFS (Point Coordination Interframe Space)** - It is used by the AP (Point Coordinator, in this case) to gain access to the medium before any other station. This value is equal to one SIFS plus one SlotTime.

- **DIFS (Distributed Coordination Function Inteframe Spacing** - It is used by a station that wishes to initiate a new transmission. It is calculated as a PIFS plus one SlotTime.

With the introduction of a TF, in the case of a collision it has an exponential backoff counter. If a STA wants to send a frame through the channel, first it senses the channel for a DIFS, it decrements a random backoff counter and when the counter reaches zero does the STA sends the frame. The backoff counter only decreases while the STA senses the channel to be idle, in the case of a busy channel, the STA waits until the channel is idle so it can proceed to decrement the counter to zero.
A DL-OFDMA operation follows these steps using the MAC layer protocol [6]:

1. For each TXOP the AP decides how many STAs can communicate, the size of each RU and indicates that in a field in the preamble of the PPDU;

2. The AP sends the data to multiple STAs in their allocated RU, this is the actual PPDU;

3. The AP requests a MU-Block ACK Request (BAR) to every STA;

4. The STAs send Block Acknowledgment (BA) back to the AP.

For a UL-OFDMA operation:

1. The AP decides which STA will ask for data to allocate RUs appropriately.

2. The AP polls data from the STA with a TF

3. The STAs respond with the PPDU;

4. The AP finalises with the BA.

2.3.1 Distributed Coordination Function (DCF)

DCF is a fundamental mechanism used in IEEE 802.11 Wi-Fi networks for channel access and contention management. It's responsible for ensuring that multiple devices can efficiently share the wireless channel without causing interference or data collisions. This access method ensures that at any time, only one device is transmitting on the same frequency. The standard algorithm is the Binary Exponential Backoff (BEB) and is used to solve problems with random access by multiple clients. In principle every node avoids collisions by using the CSMA/CA mechanism and the Request To Send (RTS)/Clear To Send (CTS) frame exchange if used [16]. There are two different modes of DCF, the basic access mechanism and the RTS/CTS mechanism, also known as four-way handshake. The basic mechanism represented in Figure 2.6 illustrates the basic access mode, this mechanism cannot circumvent the well known hidden node problem. This is where the RTS/CTS mechanisms represented in Figure 2.7 is used to reserve the channel before sending data.

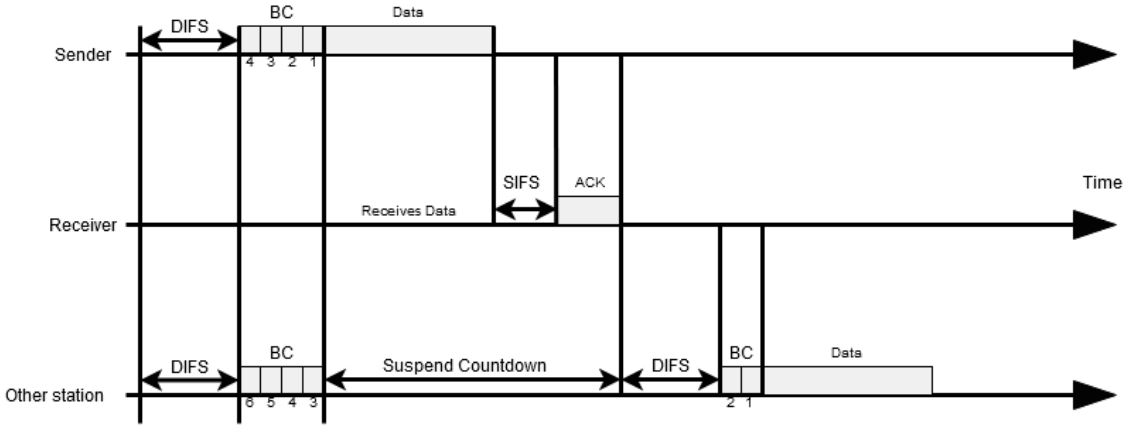


Figure 2.6: DCF Basic Mechanism adapted from [16]

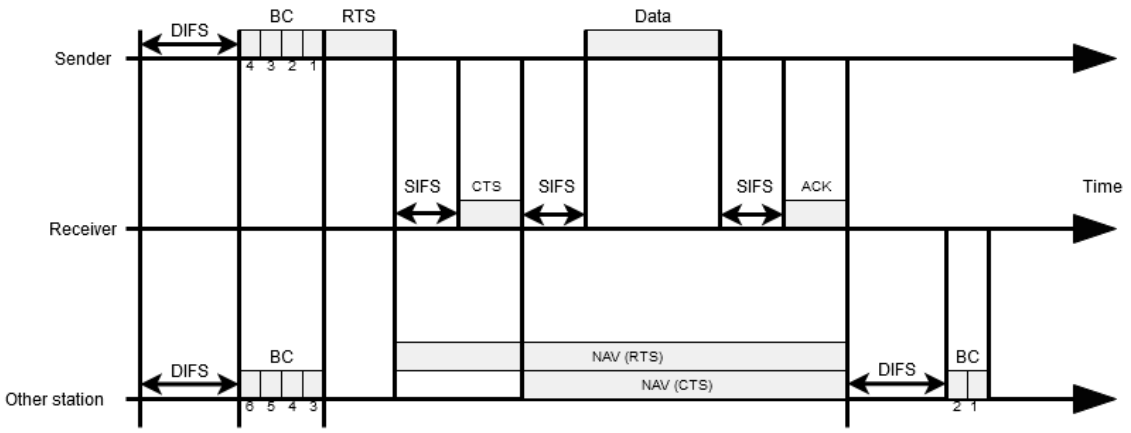


Figure 2.7: DCF with a RTS/CTS mechanism adapted from [16]

2.3.2 RTS/CTS

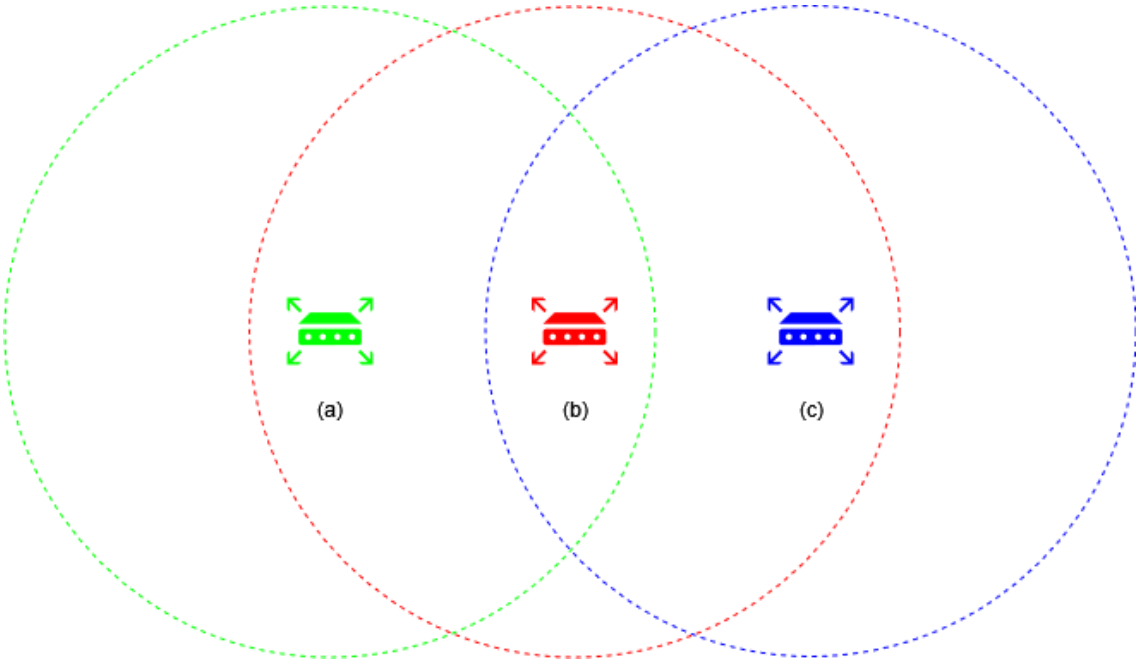


Figure 2.8: Hidden Node Problem

The Hidden Node Problem occurs when a node is not aware of the existence of another node such as in Figure 2.8. In this example node (a) is not aware of node (c) and so both of them may send data packets to (b) at the same instance, causing interference, resulting in no packet successfully reaching (b).

In order to reduce the probability of a collision between two stations, there is a mechanism defined as Virtual Carrier Sense.

The efficiency of CSMA/CA can be increased through a mechanism called RTS/CTS. This mechanism consists in sending a small RTS message to the receiver. When it arrives the receiver sends a short response with CTS. Afterwards a transmission can occur. When any station receives a RTS or a CTS, they activate their Virtual Carrier Sense also known as NAV, for the duration of the transmission. Therefore this mechanism reduces the probability of a collision occurring in the area as in the case of the Hidden Node Problem (a) requests a RTS to (b) and (c) will hear the CTS from (b) and will consider the channel as busy until the end of the active transmisson as you can see in 2.7.

2.3.3 BSS Color

The 802.11ax amendment implements a spatial reuse technique called BSS coloring. This technique is a new concept which allows for STAs to distinguish transmissions from different BSSs and quickly identify the source of a transmission. By assigning different BSS colors to adjacent BSSs, the interference between them can be reduced, thus improving spatial reuse [32] [27].

BSS color is a 6 bit parameter inside the HE SIG-A preamble and is used to distinguish frames from the same BSS or intra-BSS and Overlapping Basic Service Set (OBSS) or inter-BSS. If the BSS color is the same as the client, it is considered an intra-BSS frame otherwise it is considered an inter-BSS frame. In [28] BSS coloring simulation results show an average throughput increase per AP in UL of up to 47%.

In [14] a method that utilizes this colouring as well as proximity information to enchance the efficiency of uplink communication in WLAN environments is proposed. In this paper the physical distance between wireless devices is taken into account. By considering this proximity information, the method proposed (Proximity-based Sensitivity Control for UL (PSC-UL)) aims to intelligently allocate resources to devices further apart, maximizing the spatial reuse and reducing interference. This mechanism combines these two factors to optimize uplink transmission. By using BSS colour information to allocate different time slots for TXOP therefore minimizing interference, as well as taking advantage of the proximity information to adjust the transmission power of devices accordingly. Through their simulation and analysis the proposed approach improves the spatial reuse of the uplink channel in WLANs. By mitigating interference and allocating resources more efficiently, the method enchances the overall performance of uplink communication, leading to better throughput and reduced latency in densely populated WLAN environments.

2.3.4 Network Allocation Vector (NAV)

In normal Wi-Fi communication, the channel access uses listen-before-transmit principle, in which the station senses the medium before transmitting the frames. The 802.11ax amendment introduces two NAVs that are mandatory for a HE client, they are the basic NAV and a brand new which is the intra-BSS NAV. They are both used to protect frames from clients in dense scenarios. When a HE AP obtains a TXOP, the new intra-BSS NAV is used to prevent clients from accessing the channel. If any of the NAVs is nonzero, the medium is considered busy, otherwise the medium is considered idle. The introduction of 2 NAVs allows the usage of 1 for the station's own BSS and the other for all OBSS, this allows to change each NAV independently [13].

2.3.5 Target Wake Time

This feature which was initially introduced with the 802.11ah amendment, is a power-saving mechanism that allows an AP to manage activity in the Wi-Fi network, in order to minimize medium contention between STAs. By allocating STAs at different times, and concentrating frame exchanges in predefined periods, it is possible for STAs to remain awake for a shorter period of time in power-save mode, which in turn decreases power consumption. In 802.11ax there are two different types of TWT, the individual TWT and the broadcast TWT. In [1] it is proposed an Uplink OFDMA Random Access (UORA) grouping scheme that improves the performance of IEEE 802.11ax for ultra-dense networks. Based on a TWT grouping mechanism it can help reduce collisions in ultra-dense networks.

2.4 QoS in 802.11

One of the QoS mechanisms used in IEEE 802.11ax was introduced originally in 802.11e (released in 2005) that defined multiple mechanisms to provide QoS, mainly the Enhanced Distributed Channel Access (EDCA) and Hybrid Coordination Function Controlled Channel Access (HCCA). The main idea in EDCA is to differentiate traffic based on a priority scheme, by altering the average waiting time, the Contention Window (CW), or the IFS. Changing the CW is usually better than changing the IFS to decrease delay and increase the total throughput, because changing the IFS values only provides priorities, while differentiating the CW provides priorities and reduces collisions. Even with these changes EDCA cannot guarantee QoS, since it is based in random access [33].
The HCCA access method is a refined coordination mechanism within the EDCA framework of IEEE 802.11e, HCCA introduces a contention-free access mechanism enhancing QoS for time-sensitive applications. Parameter configuration of timing and duration of contention-free periods allows for efficient and predictable channel access for specific traffic classes. Traffic differentiation such as prioritization of traffic streams based on their QoS requirements, this is achieved by assigning different types of traffic. The HCCA also deploys a centralized polling mechanism where the AP polls STAs to transmit data in a scheduled and deterministic manner, avoiding collisions [11][25].
Dent et al [10] describe the challenges of IEEE 802.11 in dense deployment as well as some possible solutions with the implementation of 802.11ax characteristics.
Lee et al [15] proposes a UL OFDMA based Hybrid Channel Access (OHCA) procedure. With EDCA for UL-OFDMA it is possible to allow APs to prioritize channel access of STAs even if those STAs are using different 802.11 amendments, this allows for backwards compatibility [20].

3 Methodology

The purpose of this work is to evaluate on which conditions it is possible to maintain a reliable constant bitrate of 400 Mbps on a IEEE 802.11ax network in the presence of interfering traffic. This chapter presents a set of ns-3 simulations designed to evaluate the measuring error due to the simulation time considered and the achievable throughput in one interference scenario. The simulation scenarios are described in Section 3.3.

3.1 NS-3

Network Simulator ns-3 [21] is an open-source software, under GNU GPL v2 licensing, which is a discrete-event simulator, mainly targeted for research and educational use. The programming languages used for this simulator are C++ (for the Core and Model) and Python (for scripting and virtualization). The ns-3 project first started on June 2006. The version that is used in this dissertation is ns-3.34, which was released on July 14, 2021. the ns-3 consists of multiple modules and classes that interact between each other that together create a functional system. Their Application Programming Interface (API) is properly documented in the official webpage, leaving the user with all the necessary information to study and comprehend its implementation, and change it accordingly to the users necessity. For all models available in ns-3 please read the model library [22].

3.1.1 Wi-Fi Module in NS-3

The Wi-Fi module in ns-3 operates using a DCF infrastructure. The ns-3 also implements MAC Service Data Unit (MSDU) and MAC Protocol Data Unit (MPDU) aggregation. It supports DL and UL OFDMA QoS-based EDCA and queueing extensions. Multiple Propagation Loss Models, Delay models and Rate Control Algorithms are also available. The 802.11 models in ns-3 provide an accurate MAC implementation of the specification and a packet-level abstraction of different PHY models. The ns-3 *WifiNetDevice* works like a WNIC, its design can be viewed in Figure 3.1 and it's defined in the directory *src/wifi* inside the ns-3 source code.

The module design documentation [23] contains additional information about the Wi-Fi module implementation in ns-3.

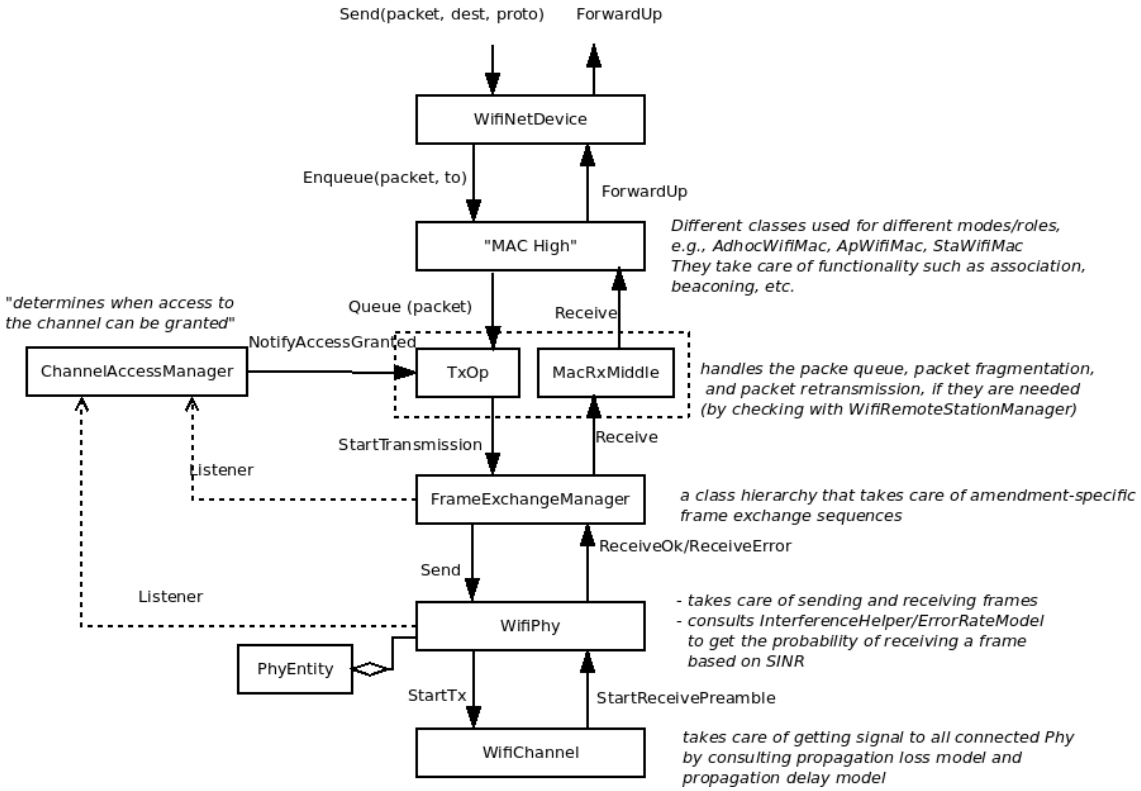


Figure 3.1: Wi-Fi Net Device Architecture in ns-3

Figure 3.2 shows the required SNR for a successful frame transmission across all MCS values. The OFDMA implementation in ns-3 for 802.11ax is validated in [18]. Models for frame detection and packet error have been validated and several examples are provided with the ns3 source code for validation of certain implementations, such as the wifi-spatial-reuse where the benefits of the OBSS Preamble Detection feature is demonstrated, among others present in the examples folder.

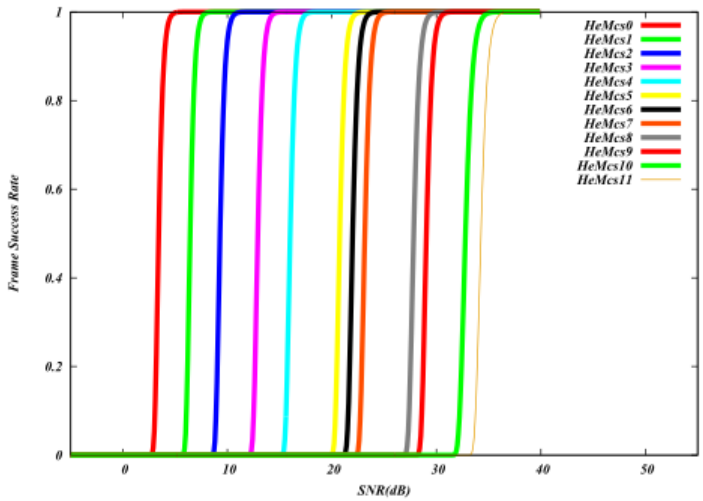


Figure 3.2: Frame Success Rate through MCS adapted from [21]

3.2 Simulation Scenarios

For this work, five scenarios were simulated using two separate C++ files. The simulations were executed in a computer using Xubuntu 20.04 with the following specs: an AMD Ryzen 7 1700 processor, with 64 GB of Random Access Memory (RAM) and an RTX2060 Super GPU.

The WiFi standard selected for all simulations was the 802.11ax at 5 GHz. The PHY model used was the Spectrum model, in particular the MultiModelSpectrumChannel. The error rate model used is the default "TableBasedErrorRateModel". Multiple rate control algorithms are available in ns-3; the one selected for all simulations was the "ConstantRateWifiManager". In practice, this means that a fixed MCS is used during the simulation, allowing us to test the influence of this parameter.
We run a first simple scenario using only 1 BSS, to evaluate the difference in throughput across the MCS, GI and Channel Width, visible in Figure 3.3. The code used for this first scenario is available in the Annex I named 1BSS.cc, and the results are present in Section 3.3.1.

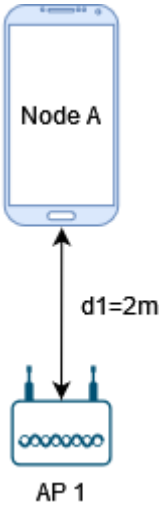


Figure 3.3: Simulation Scenario - 1BSS

For all of the other scenarios presented in Section 3.3.2 a different setup was considered, using the simulation code in the file wifi-AP-interference-test3.cc reproduced in Annex I. This scenario includes a second BSS with three additional devices, 1 AP and 2 STAs, represented in Figure 3.4 below.

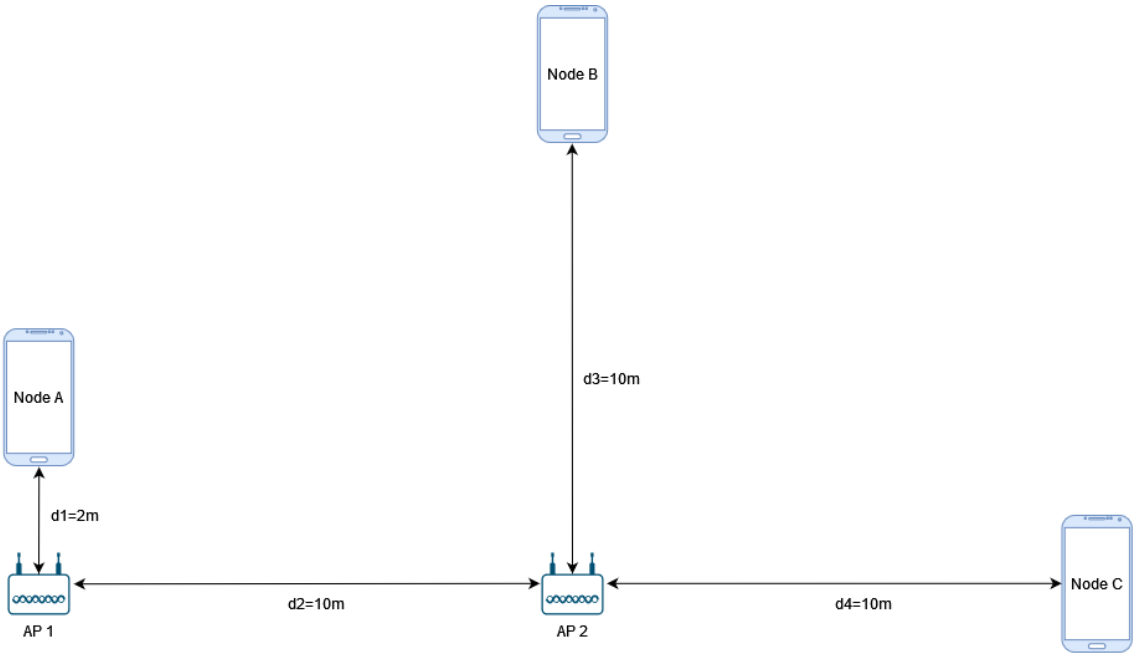


Figure 3.4: Simulation Scenario - 2BSS

For both of the scenarios, the transmission power of the STAs were *10.0 dBm*, the APs were *21.0 dBm* and the EDCA threshold was of *-62 dBm*. All of these values were selected by default.

The propagation loss models used were the default LogDistancePropagationLossModel, using an exponent of 3. Please note that the default log distance model is configured

with a reference loss of 46.6777 dB at distance of 1m.

All terminals and APs have 4 antennas, supporting MIMO transmission with 4 spatial streams for both TX and RX. The default Acknowledgment Manager used was the Aggregated MU-BAR. RTS/CTS was not activated. The MPDU buffer size is 256 bytes. The mobility model for all simulated nodes was the "ConstantPositionMobilityModel", as no node moved during the simulation. The Flow Monitor model was used to retrieve data, mainly the txPackets, txBytes, rxPackets and rxBytes of the simulation. With the parameters obtained through the Flow Monitor model the actual data rate or input offered in Mbps, the obtained throughput in Mbps and the number of lost packets is calculated by using:

$$DataRate = \frac{txBytes * 8}{1000 * 1000 * duration}, \tag{3.1}$$

$$Throughput = \frac{rxBytes * 8}{1000 * 1000 * duration}, \tag{3.2}$$

$$LostPackets = txPackets - rxPackets, \tag{3.3}$$

where *rxBytes* is the amount of Bytes registered by the Flow Monitor as successfully received, *txBytes* the amount of Bytes transmitted, *duration* is the simulation time in seconds, *txPackets* and *rxPackets* the number of packets sucessfully transmited and received.

3.3 Results

In this section the results and discussion are presented for all five simulations. Additional results, such as all samples retrieved from all 5 scenarios were added to Appendix B as tables.

3.3.1 First Iteration

As an initial scenario, a single simulation was executed in a loop that included twelve MCS values for three GI values with two different channel widths. In total 72 simulations were executed. In this scenario the application used to initiate the communication within the program was a simple *UdpServerHelper* and *UdpClientHelper* which sent 10 000 packets at a rate of 1 packet per 0.00001 second, each packet with 1472 Bytes of payload size. Each simulation took approximately 5 minutes to execute. The results for these 72 simulations are represented visually as a Scatter Plot in Figure 3.5 and the data in Table 3.1.

Figure 3.5: Throughput Across MCS for a GI of 800,1600,3200 ns and a Channel Width of 80,160 MHz

Table 3.1: Throughput across MCS, Guard Interval and Channel Width

		Throughput ($Mbit/s$)	
MCS	Guard Interval (ns)	80 MHz	160 MHz
0	3200	81.0656	154.521
	1600	89.8458	170.081
	800	94.8519	178.988
1	3200	154.691	281.713
	1600	170.237	307.636
	800	179.217	322.234
2	3200	222.087	392.508
	1600	243.42	425.88
	800	255.662	444.409
3	3200	282.312	479.813
	1600	308.272	517.974
	800	323.026	538.812
4	3200	392.431	630.312
	1600	425.665	674.17
	800	444.32	697.441
5	3200	479.715	738.036
	1600	517.698	784.223
	800	538.714	808.461
6	3200	521.247	796.357
	1600	561.155	818.935
	800	583.084	818.962
7	3200	560.06	818.624
	1600	601.416	818.95
	800	623.953	818.949
8	3200	630.202	818.629
	1600	673.819	818.95
	800	697.306	818.965
9	3200	672.249	818.638
	1600	717.015	818.949
	800	740.888	818.967
10	3200	720.391	818.633
	1600	765.923	818.964
	800	790.222	818.953
11	3200	756.305	818.645
	1600	802.619	818.966
	800	818.771	818.975

Analysing the results we can conclude that, a higher MCS and a larger channel width provide a higher throughput. Across all MCS values, a shorter GI also provides an increase in throughput. Starting at a MCS value of 6 for the 160 MHz, we reach the physical limitations that were declared initially, at approximately 818 Mbps, while this value limit is only achieved at an MCS of 11, for the channel width of 80 Mhz. Unfortunately, for this first scenario the simulation was only executed once, and the sample size is of only one simulation for each value.

3.3.2 Interference Scenario

The preparation of an interference scenario for testing the Catheter Lab has the objective of mixing in the same radio channel signals from two APs and terminals associated. They represent the device that performs the examination and transmits the results which is Node A, the terminal that receives the traffic from this node, AP 1, two additional interfering Nodes that could represent the cellphones connected to a local Wi-Fi Network inside the building which would be AP 2. During the preparation of the scripts we identified several limitations and bugs of the ns-3 implementation considered:

1. Freezing Simulations: The simulation kept freezing when left running overnight, for more than 8 hours, the simulation never finished even for low data rates and simulation times. To overcome this issue, a new implementation was developed. For this new implementation the new application for communicating was changed to an *onOffHelper* where instead of selecting the number of packets to send, and after sending them the simulation stops, this application starts to send data in a Constant Bit Rate (CBR) stream that saturates the channel, when the time value *StartTime* is reached and ends when the *StopTime* is reached. The payload for all nodes in this communication is of 1500 Bytes, the data rate for Node A is of 400 Mbps.

2. Colliding Beacons: Initially, the script contained some errors, where we encountered colliding beacons, the default configuration of an AP for beacon generation is the periodic transmission of the beacon. When we added both AP 1 and AP 2, they were sending periodic beacons exactly at the same time, always leading to collisions and consequently the APs never started. By enabling beacon jitter, the APs send their beacon at different instances and we were able to overcome this issue.

3. ARP failures: In ns-3.34 the Address Resolution Protocol (ARP) does not work perfectly, therefore a workaround was implemented. By first starting the communication with a *UdpEchoClientHelper* and sending 3 pings we were able to circumvent this issue, and all nodes were able to start and begin their communication with the CBR stream.

With the new, working code, four different scenarios were simulated. Some values are common to all simulations, such as BSS coloring was assigned per BSS and the "ConstantObssPdAlgorithm" was used taking advantage of the 802.11ax spatial reuse feature with a default threshold value of -72 dBm. The MCS selected was 11, for the best possible throughput, the channel width selected was 160 MHz for the same reason, at a central frequency of 5250Hz. The GI was of 800 ns, given the small distance between the catheter transmitter and the receiver, the smaller GI value can be used for the scenario considered in the Catheter Lab.
The four interference scenarios were tested in this chapter with different goals:

1. The first set of experiments tested the influence of simulation duration in the measured Catheter Lab traffic, with the goal of defining this parameter for the other simulations. The results of this scenario can be found in Section 3.3.2.1. In this solo scenario, where a single node is transmitting without interference, the Data Rate for Node B and C were set to 0 and their respective transmission powers as well as the transmission power of AP 2 were all set to 0. The duration or simulation time was incremented by 0.1 s, starting at 0.1 s and ending at 1.0 s.

2. The second set of experiments considered a low interference scenario. In this case, the Data Rates for Node B and C were set to 10 Mbps, and their transmission power was restored to the default values. This experiment measured how the interfering traffic affected the Catheter Lab throughput. The results are present in Section 3.3.2.2.

3. The third set of experiments analyses the influence of distance in the Catheter Lab throughput, with increments of 1 meter between Node A and AP 1, increasing the value of *d1* from the initial value of 2 meters to 10 meters. The results can be found in Section 3.3.2.3.

4. Lastly, the fourth set of experiments analyses the influence of a growing interfering traffic on the Catheter Lab throughput. For this simulation, the Data Rate for Node B and C was incremented, starting from 100 bps and reaching up to 100 Mbps. The results can be found in Section 3.3.2.4.

We executed multiple bash scripts for every scenario, each one to obtain multiple samples for each simulation value. The script ran in a loop and saved relevant information such as the number of packets, bytes, and throughput to multiple text files. Subsequently, the text files were merged into a Comma-Separated Values (CSV) file that included all samples for a specific simulation. All of the data collected from the simulations is presented in tables in Appendix B. The graphics found in this chapter and in Appendix A were generated by a Python script using the CSV files. For all of the code used, please refer to Annex I.

3.3.2.1 Solo Simulation

This scenario measures the influence of simulation duration with the goal of defining this parameter for the other simulations. It contains a single node transmitting without interference. The simulation time was incremented by 0.1 s, starting at 0.1 s and ending at 1.0 s.

For a value of 0.1 s of simulation time, it took approximately 15 minutes to simulate a single sample. A sample for a value of 1.0 s of simulation time took 2 hours 7 minutes and 30 seconds to compute. The rest of the samples followed the same pattern of 12 minutes

and 30 seconds per 0.1 s of simulation time with an additional 2 minutes and 30 seconds as a base value of simulation time, this behavior is visible in Figure 3.6.

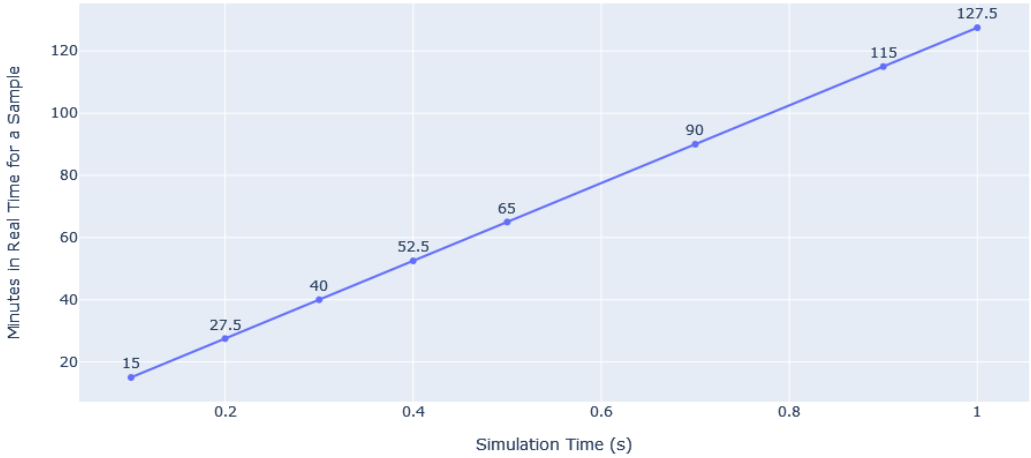


Figure 3.6: Real Time to compute a sample vs Simulation Time

Some samples for this scenario did not ended, namely the 0.6 s and the 0.8 s. The machine when executing the simulation would enter a freeze state. When left overnight for over 8 hours, the simulation simply would not produce any output. This occurred for some Random Number Generation (RNG) values for other values, happening more frequently with longer simulation time.

Table 3.2 contains the number of samples computed for each simulation and some collected results, namely the number of packets transmitted, the average number of packets received, the offered data rate and the average obtained throughput.

Table 3.2: Solo Simulation Values

				Megabits ($Mbps$)	
Time (s)	# Samples	Tx Packets	Rx Packets	Tx Offered	Throughput
0.1	100	3333	3324	407.426	406.34
0.2	100	6666	6657	407.426	406.89
0.3	42	9999	9989	407.426	407.00
0.4	40	13333	13324	407.456	407.17
0.5	25	16666	16655	407.450	407.19
0.7	60	23333	23324	407.461	407.30
0.9	20	29999	29990	407.453	407.33
1.0	30	33333	33325	407.463	407.37

In Figure 3.7, the number of transmitted, received and lost packets are presented over the simulation time. We can observe that the number of transmitted and received packets

increase linearly with simulation time by a factor of 3333 packets per 0.1 s of simulation time. The number of lost packets remains almost the same regardless of simulation time, because of the implementation used, some packets that are still in transit when the simulation stops are lost/not counted.

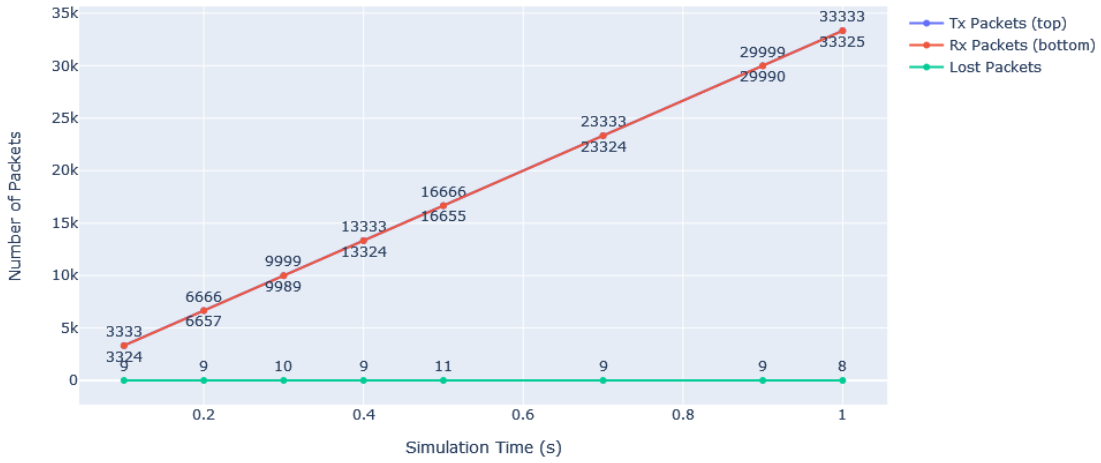


Figure 3.7: Transmitted, Received and Lost Packets Over Simulation Time without Interference

The evolution of the measured throughput for different simulation times is depicted in Figure 3.8 and in Figure 3.9. The longer the simulation time, those lost/unaccounted packets are more negligible, providing a higher overall throughput result.
Figure 3.8 plots a zoomed vertical scale, to allow visualising the small throughput variations that occur with the solely variation of the simulation time. In Figure 3.8 this behavior is visible with an increase of 1 Mbps of average throughput between 0.1 s and 1.0 s of simulation time, while the offered data rate remains the same.

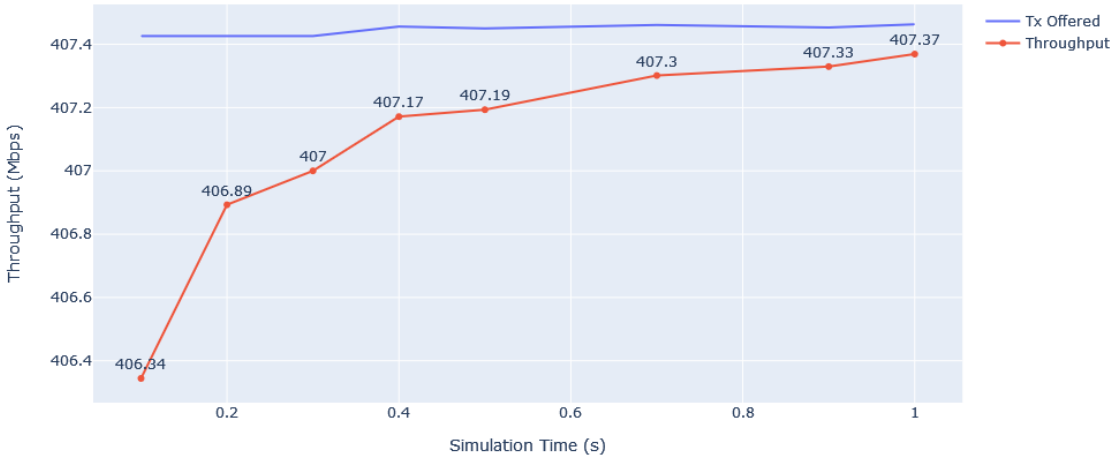


Figure 3.8: Average Throughput over Simulation Time without Interference

In Figure 3.9 the variance of throughput between samples is larger for a smaller simulation time. By increasing simulation time we can reduce this variance.

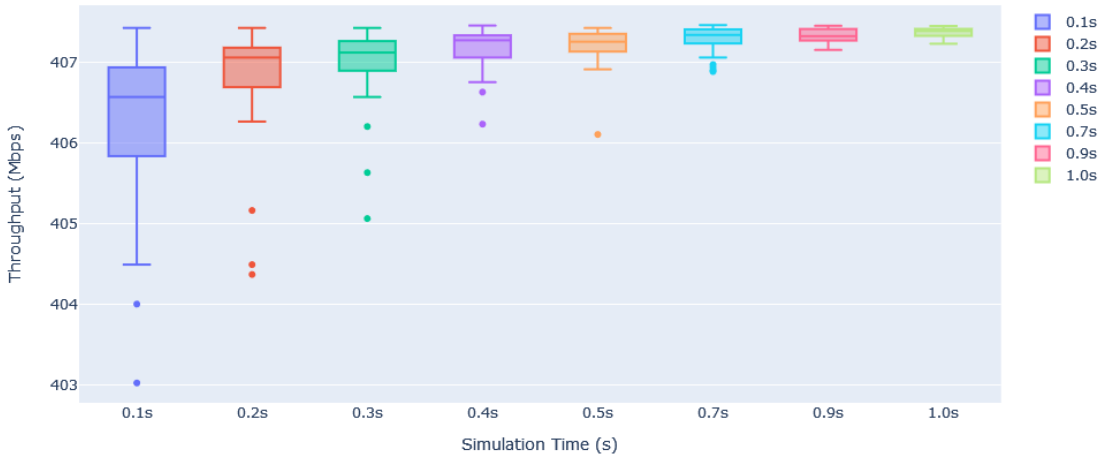


Figure 3.9: Box plot of Throughput over Simulation Time without Interference

The highest throughput achieved were perfect transmissions, where 0 packets were lost, in those cases the obtained throughput is equal to the offered Data Rate. All simulation values achieved at least one perfect simulation. The lowest throughput achieved was a sample of 0.1 s where the obtained throughput was of 403.03 Mbps, with a loss of 36 packets. This corresponds to a loss of 1.08% of the transmitted packets.

Considering the real time to compute a sample from Figure 3.6, and the obtained average throughput from Figure 3.8, we can observe and conclude that increasing the simulation time has a slight although almost negligible improvement on throughput. The 0.1s interval provides a faster simulation, but imprecise results, with a very high measured variance and worse results, due to the lost/unaccounted packets with much smaller transmitted packets. The 0.5 s and 1.0 s interval took to long to simulate, although with a smaller measured variance. The 0.2 s interval, provides a trade off simulation with an acceptable accuracy and an acceptable measured variance. Since each 0.2 s simulation took almost 28 minutes to execute and the 1.0 s simulation took more than 2 hours for a single sample, future simulations were considered to use 0.2 s for simulation time as a baseline.

3.3.2.2 Interference Across Time

This scenario considers a low interference scenario. The Data Rates for Node B and C were set to 10 Mbps, and their transmission power was restored to the default values. This experiment measures the interfering traffic affected the Catheter Lab throughput. When the simulation time was longer than 0.2 s the same freezing issue ensued and unfortunately occurred more often with higher simulation times. By adding additional data rate from other nodes, the simulation complexity is increased.

In Table 3.3 are the number of samples for each simulation and some collected results, such as the number of packets transmitted, the average number of packets received, the offered data rate and the average obtained throughput, for node A, B and C.

Table 3.3: Throughput Over Simulation Time with Interference

		Node A			Node B anc C		Throughput (*Mbps*)		
Time (s)	# Samples	Tx Packets	Rx Packets	Tx Offered	Tx Packets	Tx Offered	Node A	Node B	Node C
0.01	100	333	287	407.059	7	8.56	350.88	7.14	7.69
0.1	12	3333	3252	407.426	82	10.02	397.59	9.93	9.84
0.2	11	6666	6628	407.426	165	10.08	405.13	9.93	10.06
0.5	6	16666	16626	407.450	415	10.15	406.47	10.13	10.13
1.0	3	33333	33262	407.463	832	10.17	406.60	10.14	10.15

For Figure 3.10, Figure 3.11 and Figure 3.12, the results are almost identical when comparing with the solo simulation. We can observe that the number of transmitted and received packets increase linearly with simulation time by the same factor for node A and in the case for node B and C the factor is approximately 83 packets per 0.1 s. The number of lost packets remains almost the same regardless of simulation time. Although there is a higher number of lost packets when comparing between node A with interference and without interference.

Therefore, those packet were lost during the beginning of the simulation. This is possibly related to not having the ARP table initialized, which in ns-3.34, leads to dropped packets at the Internet Protocol (IP) layer. When the ARP tables are initialized, the packets start to flow to the AP almost without losses.

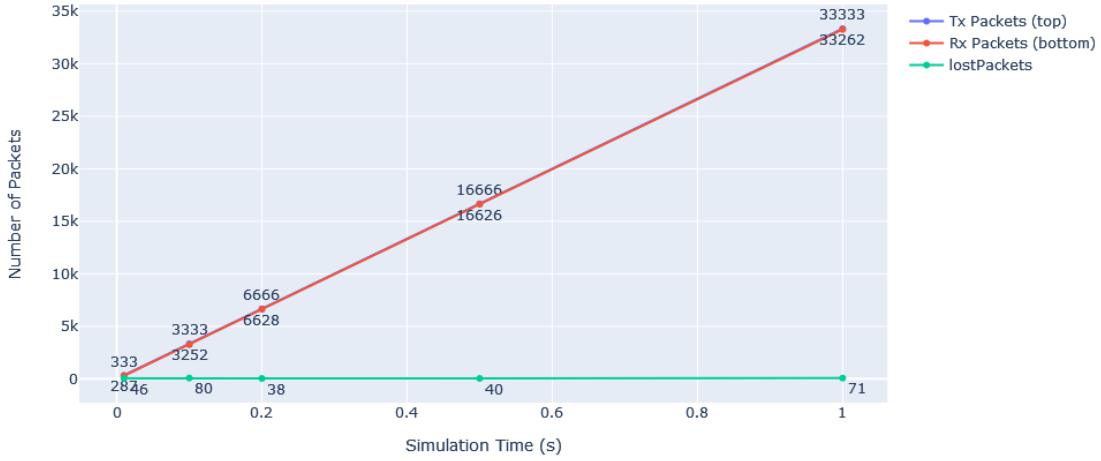


Figure 3.10: Transmitted, Received and Lost Packets for Node A across Simulation Time

For all of the simulation times there was an increase on the average of lost packets when comparing to the scenario without interference. For the 0.1 s sample there was an increase of 71 lost packets, which results in a total packet loss ratio of 2,5%. For the 0.2 s sample there was an increase of 29 lost packets, which results in a total packet loss ratio of 0,6%. For the 0.5 s sample there was an increase of 29 lost packets, which results in a total packet loss ratio of 0,25%. For the 1.0 s sample there was an increase of 63 lost packets, which results in a total packet loss ratio of 0,21%.

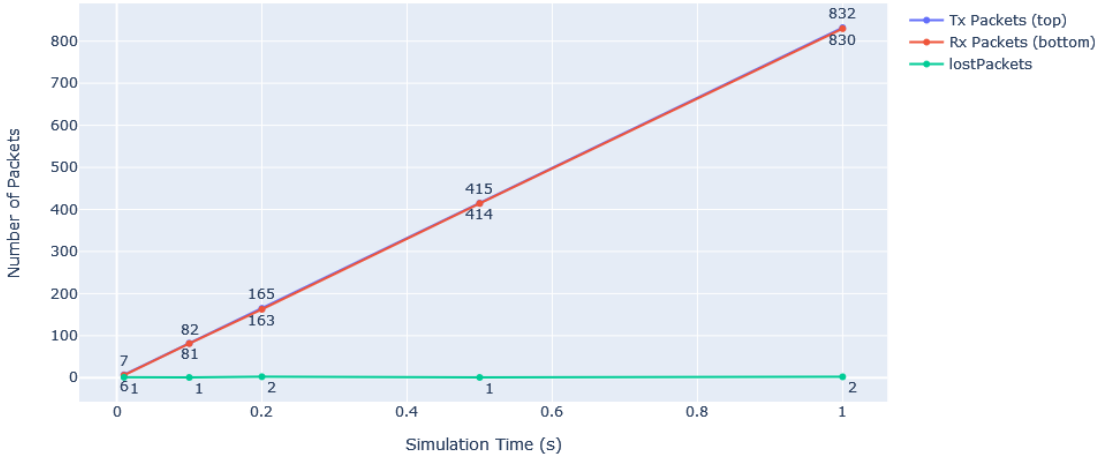


Figure 3.11: Transmitted, Received and Lost Packets for Node B across Simulation Time

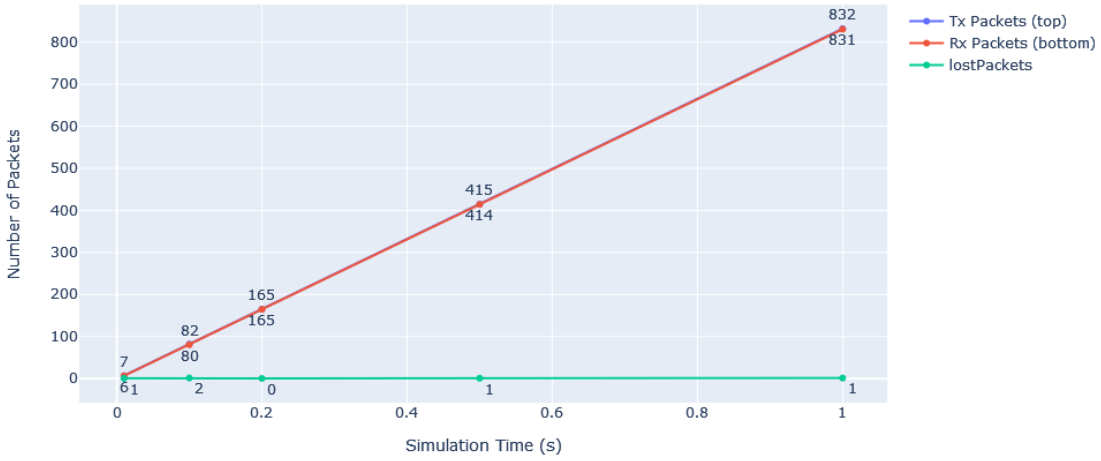


Figure 3.12: Transmitted, Received and Lost Packets for Node C across Simulation Time

Figure 3.13 similarly to the solo simulation depicts the throughput of Node A with 2 interfering nodes of 10 Mbps each, for different simulation times. Figure 3.14 and Figure 3.15 depict the throughput of Node B and C. We can observe and conclude that increasing the simulation time has a slight although negligible improvement on throughput as well. The 0.01s interval provides very fast simulations, but imprecise results, with a very high measured variance and provided significantly worse results. The 0.5 s and 1.0 s interval took to long to simulate, although with a smaller measured variance when freezing did not occur. The 0.2 s interval, provides a trade off simulation with an acceptable accuracy and an acceptable measured variance.

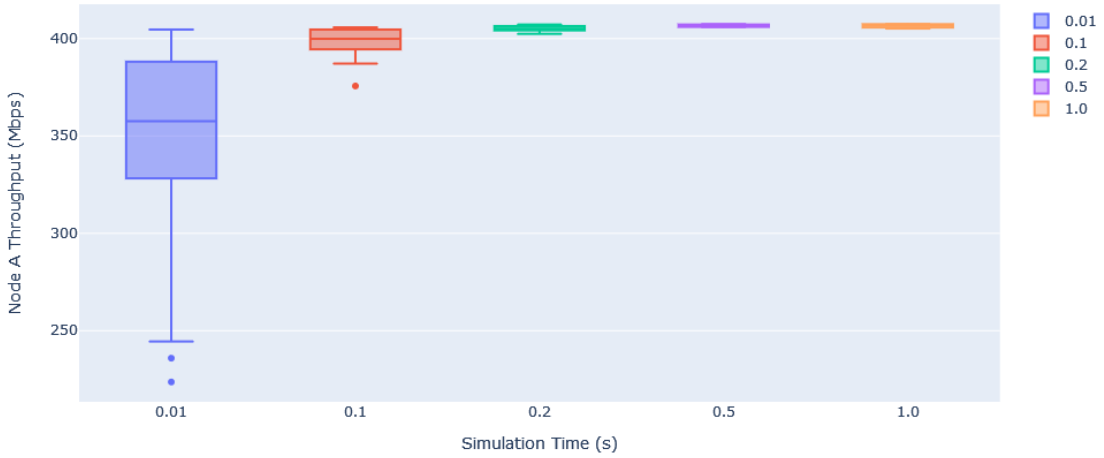


Figure 3.13: Throughput of Node A over Simulation Time with Interference

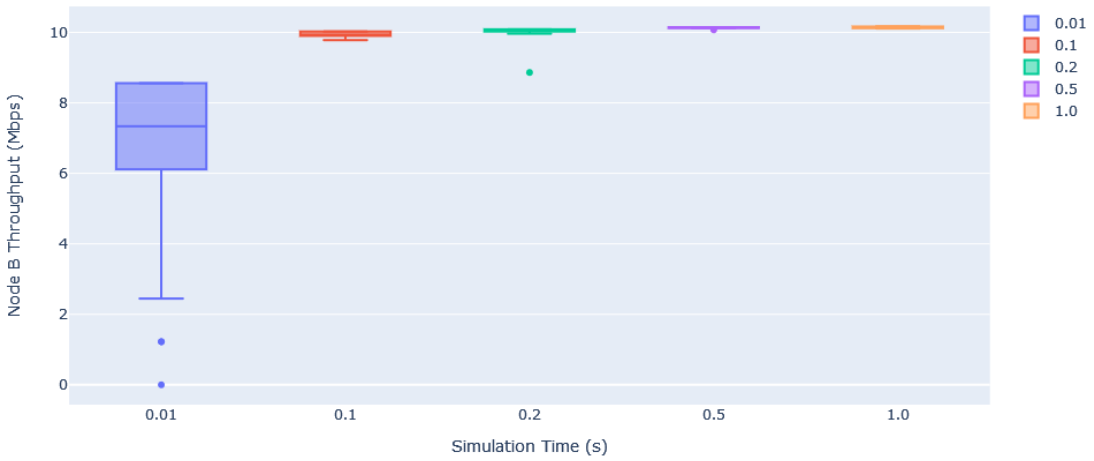


Figure 3.14: Throughput of Node B over Simulation Time with Interference

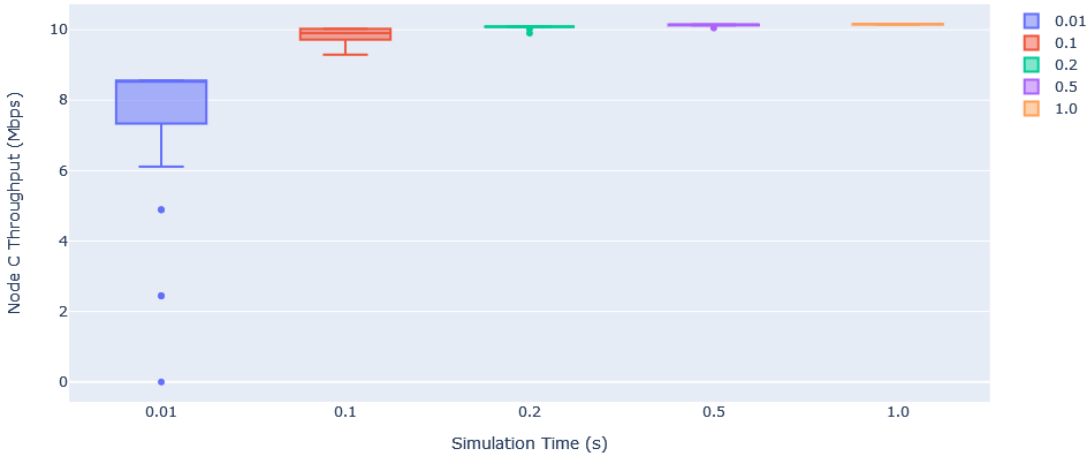


Figure 3.15: Throughput of Node C over Simulation Time with Interference

3.3.2.3 Interference Across Distance

The third scenario, analyses the influence of distance in the Catheter Lab throughput. The distance between Node A and AP 1 was incremented by 1 meter, starting with an initial value of 2 meters to 10 meters. Table 3.4 contains some of the collected results as well as the number of samples for this simulation.

Table 3.4: Throughput across Distance

		Average Throughput ($Mbps$)		
Distance (m)	# Samples	Node A	Node B	Node C
2	10	405.30	10.04	10.06
3	10	405.22	9.92	10.06
4	10	405.22	9.92	10.06
5	10	405.22	9.93	10.06
6	10	405.22	9.93	10.06
7	10	405.22	9.93	10.06
8	10	405.22	9.93	10.06
9	10	405.22	9.93	10.06
10	10	405.22	9.93	10.06

In Figure 3.10 we can conclude that for small distances there is no increase in the number of lost packets.

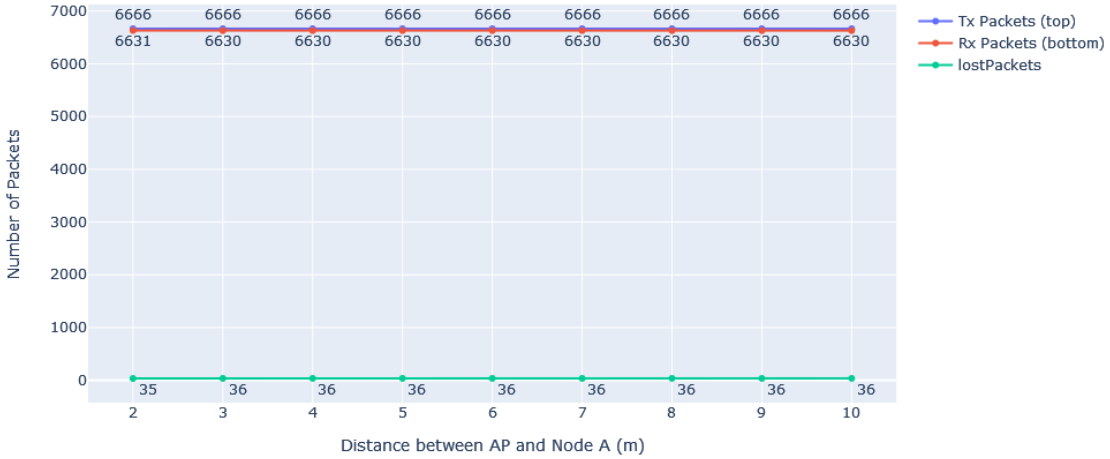


Figure 3.16: Transmitted, Received and Lost Packets for Node A across Distance

By observing 3.17 we can conclude that between 3 and 10 meters, there is no difference in variance of obtained throughput. For all of these values, the highest throughput obtained was of 407.181 Mbps and the lowest value obtained 402.414 Mbps, this results in a variance of 1,2%.

For 2 meters we have a slightly higher average throughput, although with a higher level of variance. The highest throughput obtained was also 407.181 Mbps and the lowest obtained throughput for this distance was of 400.336 Mbps, this results in a variance of 1,7%.

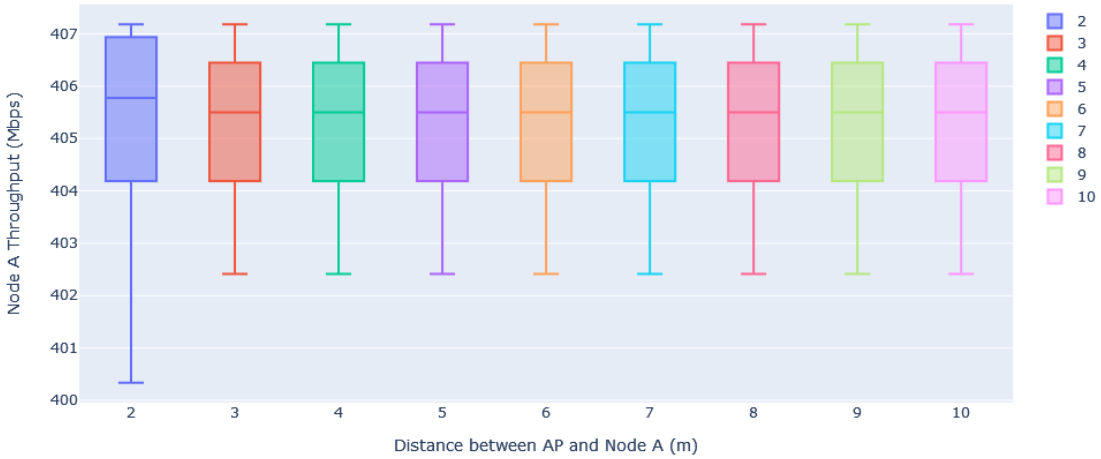


Figure 3.17: Throughput of Node A across Distance

3.3.2.4 Incremental Interference

For our final simulation scenario, all of the simulations were done with a simulation time of 0.2 s and the distances in Figure 3.4. The data rate of the interfering nodes was increased starting with 100 bps and reaching up to 100 Mbps. The table 3.5 includes the number of samples for each simulation as well as the average throughput calculated for each Node. Node B and C only begin transmitting when the data rate is 100 000 Bytes or higher.

Table 3.5: Increasing Interference Values

		Node B anc C		Throughput ($Mbps$)		
Interfering Data Rate	# Samples	Tx Packets	Tx Offered	Node A	Node B	Node C
100	100	N/A	N/A	406.89	N/A	N/A
1 000	100	N/A	N/A	406.89	N/A	N/A
10 000	100	N/A	N/A	406.89	N/A	N/A
100 000	100	1	0.06	406.95	0.06	0.06
1M	91	16	0.98	406.92	0.98	0.98
2M	99	33	2.02	406.18	1.99	1.99
5M	50	82	5.01	406.23	5.00	5.00
10M	88	165	10.08	404.04	10.02	10.04
20M	28	331	20.23	394.12	20.10	20.12
30M	15	497	30.38	391.80	29.90	30.22
40M	15	663	40.52	375.86	40.17	40.08
50M	15	829	50.67	379.89	50.12	50.04
60M	15	995	60.81	376.18	60.15	59.88
70M	15	1160	70.90	379.40	70.18	70.43
80M	15	1326	81.05	370.19	80.60	80.30
90M	15	1492	91.19	363.39	90.40	90.38
100M	40	1658	101.34	361.43	100.40	100.18

Due to the large number of values for interfering data rate, and the difference in scale of the data rate, the results were split into two parts. This way the data is easier to visualize and analyze.

The first part starts with 100 bps of interference and ends with 10 Mbps following the Table 3.5 values. In Figure 3.18, Figure 3.19 and Figure 3.20 are the transmitted, received and lost packets for this part of the simulation for each node. We can observe a slight increase in the number of lost packets in Node A starting with 2 Mbps per interfering node, or 4 Mbps of total interference. For Node B and C we can observe that they begin sending packets with an offered data rate of 100 000 Bytes.

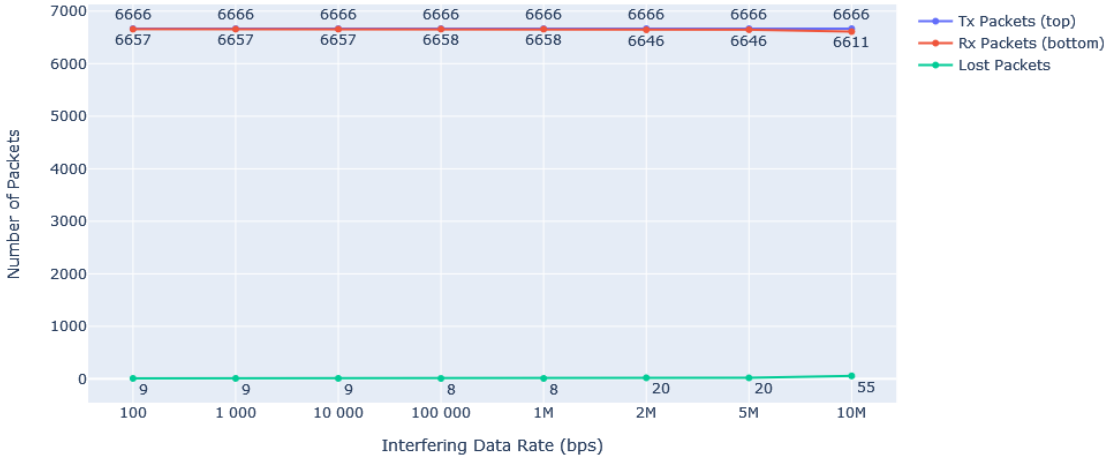


Figure 3.18: Transmitted, Received and Lost Packets for Node A with Increasing Interference Part 1

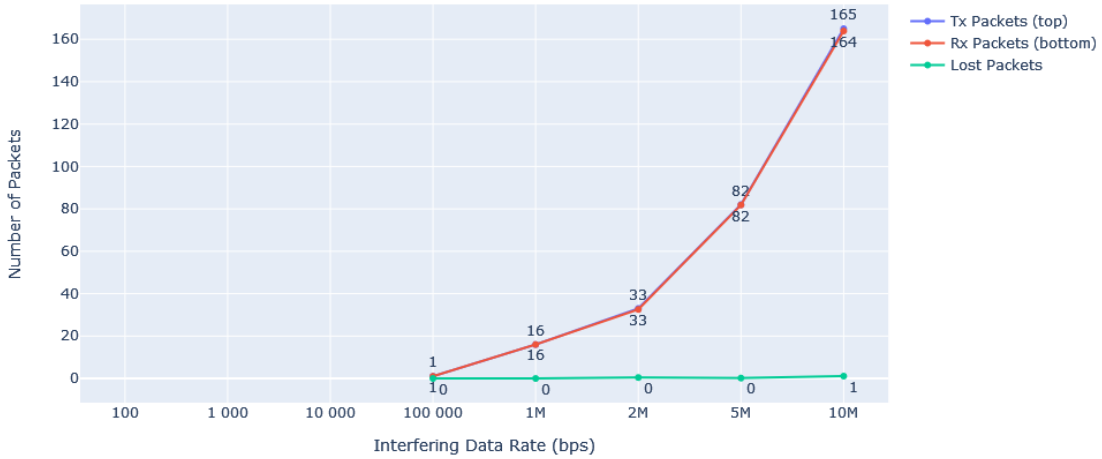


Figure 3.19: Transmitted, Received and Lost Packets for Node B with Increasing Interference Part 1

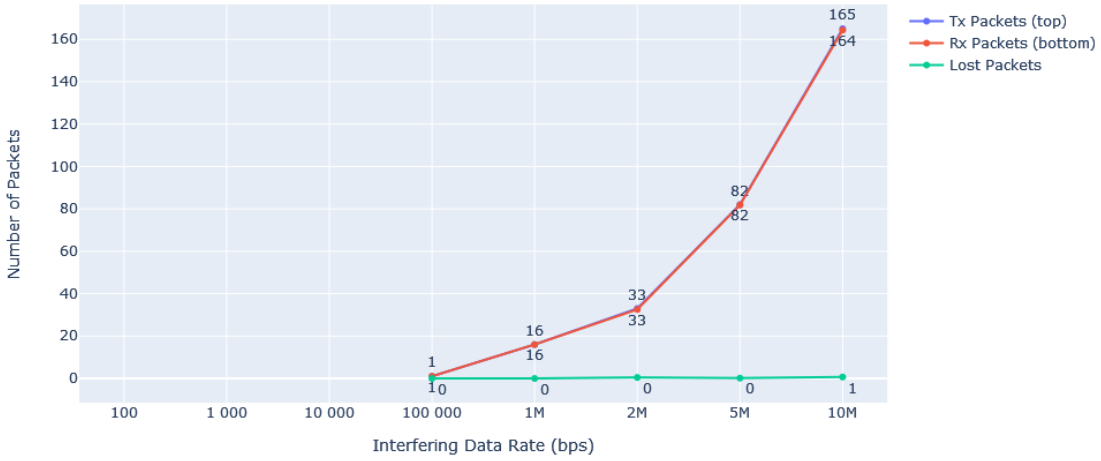


Figure 3.20: Transmitted, Received and Lost Packets for Node C with Increasing Interference Part 1

In Figure 3.21 the average throughput of Node A is depicted. Here we can see a decrease in obtained throughput starting at 1 Mbps. At 10 Mbps of interference per node there is a decrease of 1% of throughput when comparing to the solo scenario without interference. For 5 Mbps of interference per node there is a slight increase in average throughput, this is due to the lower number of samples.

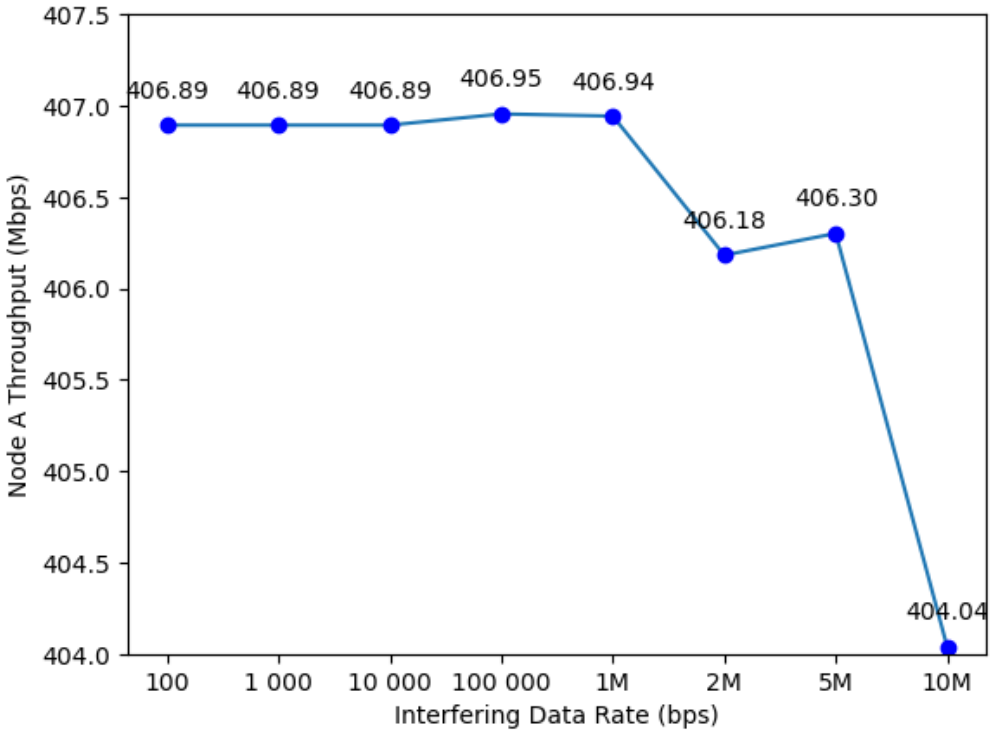


Figure 3.21: Average Throughput of Node A with Interference 1-10 Mbps

Figure 3.22 also depicts the throughput of Node A, for 10 Mbps of interfering data rate the variance of the throughput is higher while having a lower average throughput. The smallest sample obtained is of 385.789 Mbps which is a decrease of 5% in throughput.

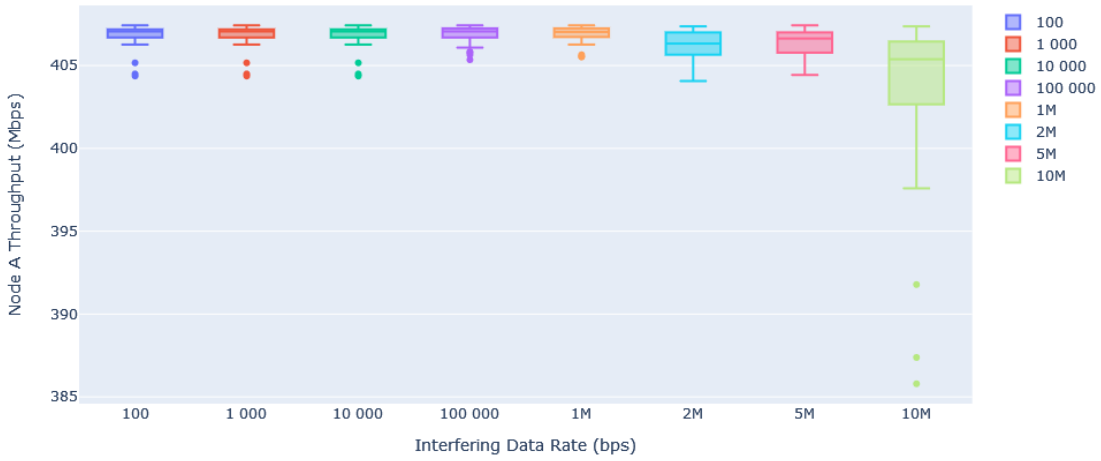


Figure 3.22: Node A Throughput with Interference 1-10 Mbps

In Figure 3.23 and Figure 3.24 the throughput of Node B and C is visible, with a linear

increase of average throughput with the offered data rate.

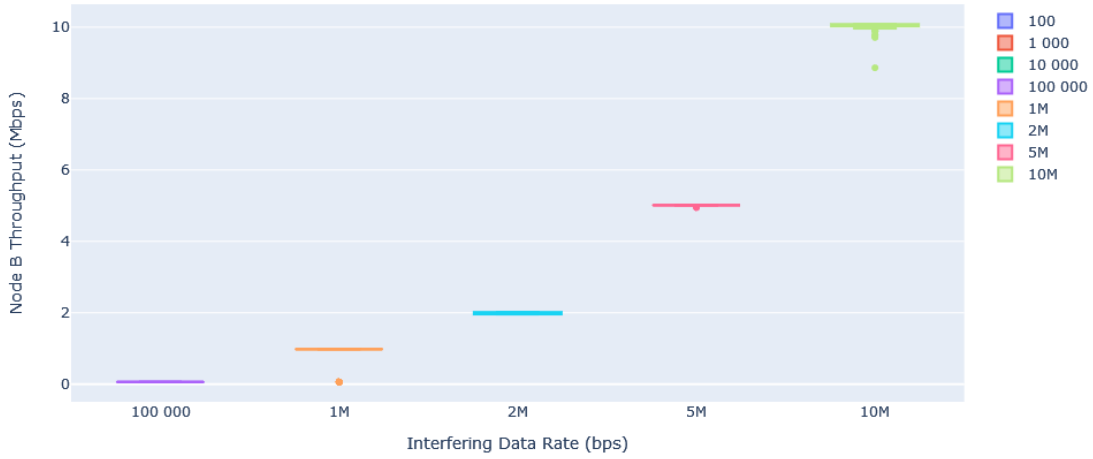


Figure 3.23: Node B Throughput with Interference 1-10 Mbps

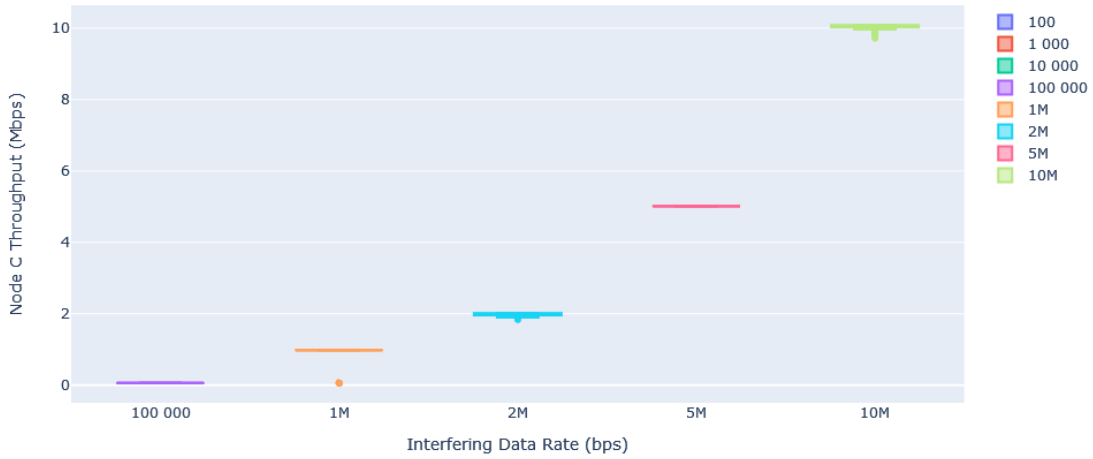


Figure 3.24: Node C Throughput with Interference 1-10 Mbps

The second part of this simulation starts at 10 Mbps and, by increments of 10 Mbps, ends with an interfering data rate of 100 Mbps, also using the values in Table 3.5. Figure 3.25, Figure 3.26 and Figure 3.27 show the number of packets transmitted, received and lost across the interfering data rate. We can see a decreasing linear approximation in the number of received packets for Node A and an increase in the number of lost packets. For

100 Mbps of interfering rate, there is a 11% packet loss. For Node B and C, the number of sent packets increased linearly with the offered data rate by a factor of approximately 166 packets sent per 10 Mbps of offered data rate.

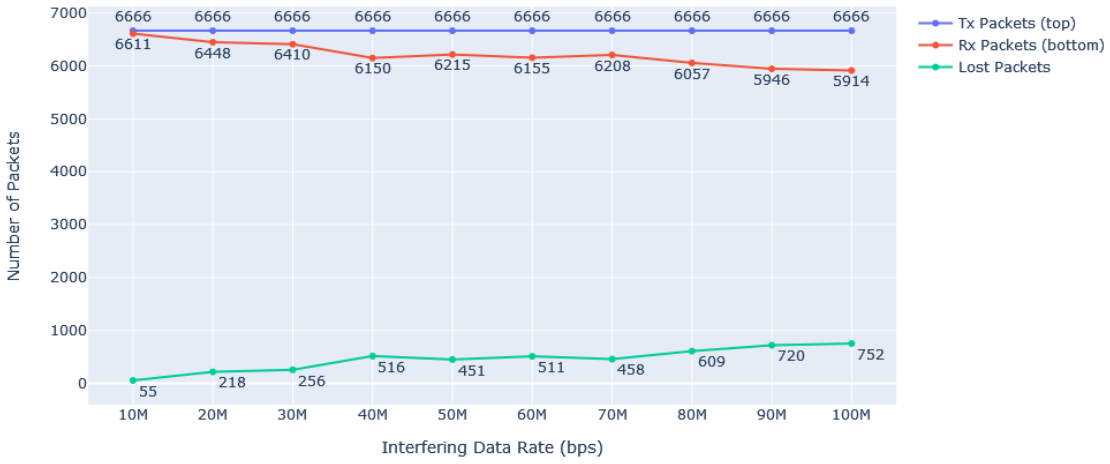


Figure 3.25: Transmitted, Received and Lost Packets for Node A with Increasing Interference Part 2

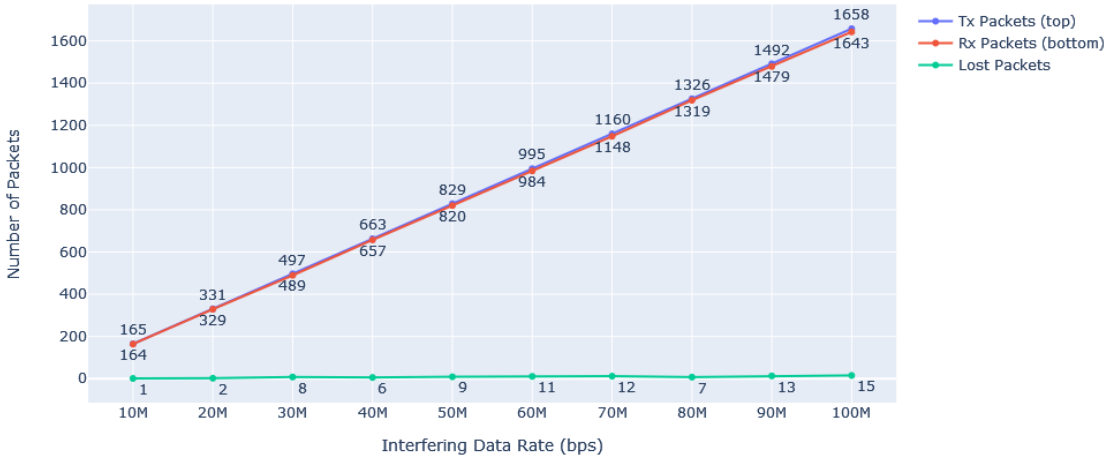


Figure 3.26: Transmitted, Received and Lost Packets for Node B with Increasing Interference Part 2

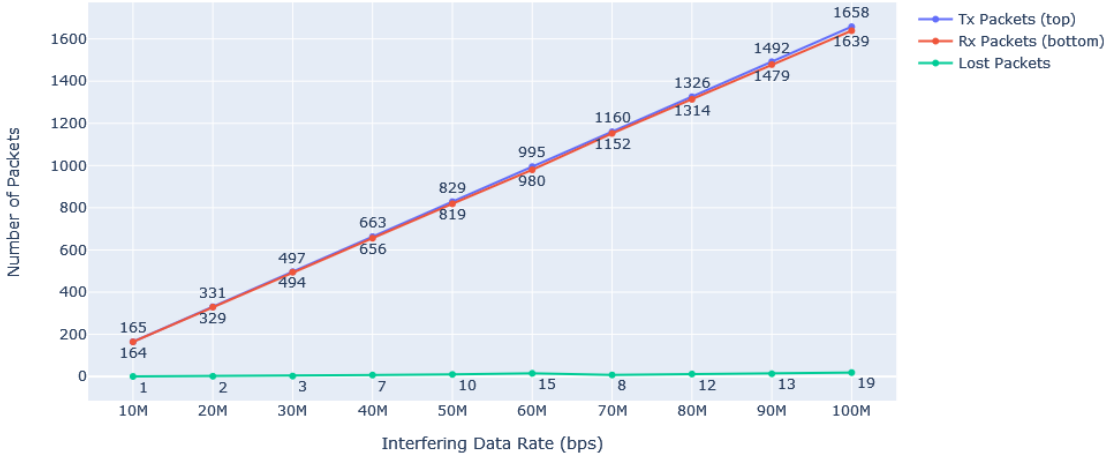


Figure 3.27: Transmitted, Received and Lost Packets for Node C with Increasing Interference Part 2

In Figure 3.28 the average throughput of Node A is depicted. Here we can clearly see a decrease in obtained throughput from 10 Mbps until 100 Mbps of interfering data rate. At 100 Mbps of interfering data rate, the average throughput decreases by almost 12% when compared with no interference.

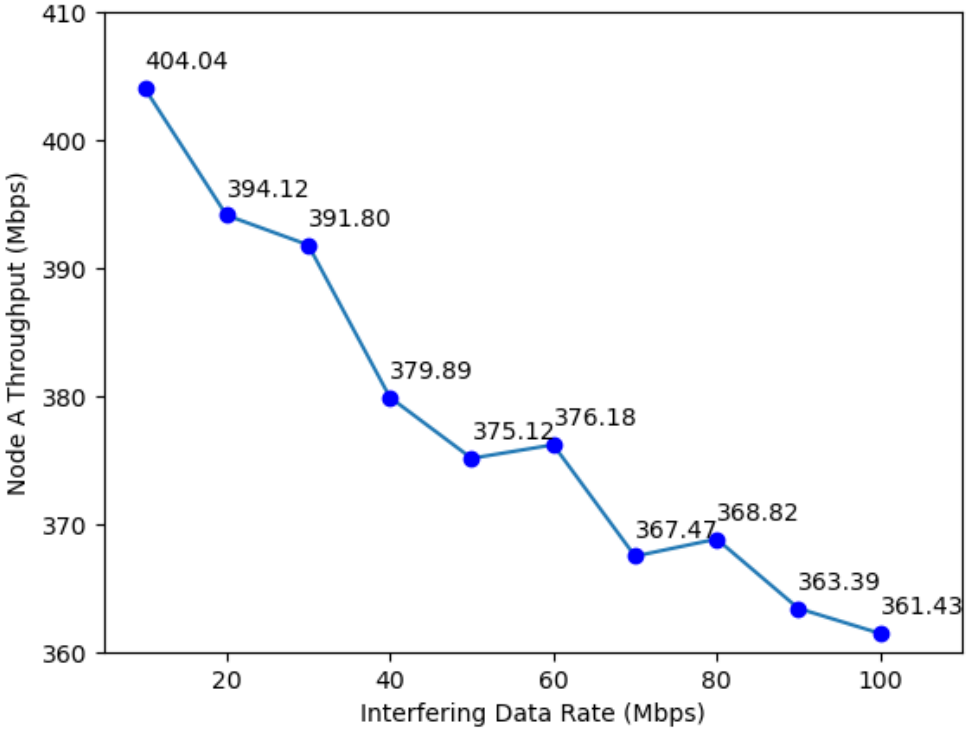


Figure 3.28: Average Throughput of Node A with Interference 10-100 Mbps

Figure 3.29 contains the throughput of Node A. We can see that the higher interfering data rate increases the variance of obtained throughput for Node A, having the lowest value of throughput of 281 Mbps. This is a significant decrease of 30% of obtained throughput.

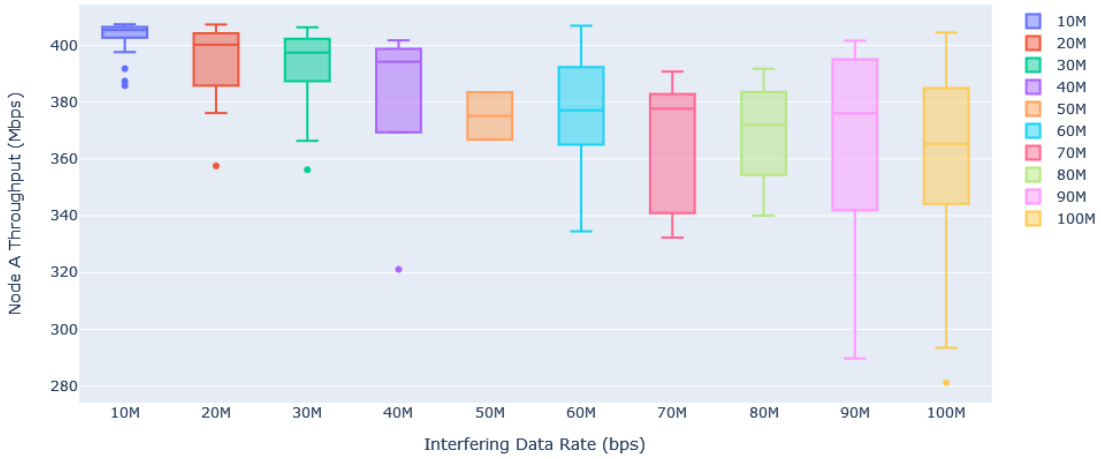


Figure 3.29: Node A Throughput with Interference 10-100 Mbps

In Figure 3.30 and Figure 3.31 the throughput of Node B and C can be observed as well. The throughput of Node C is identical to Node B and increases linearly with the offered data rate. The smallest value of throughput obtained for node B is 98.77 Mbps which is a decrease of 2,5% in throughput. For node C the lowest obtained throughput is 87.89 Mbps which is a decrease of 13,22% in throughput. This discrepancy between Node B and Node C might be due to the low number of samples and a short simulation time.

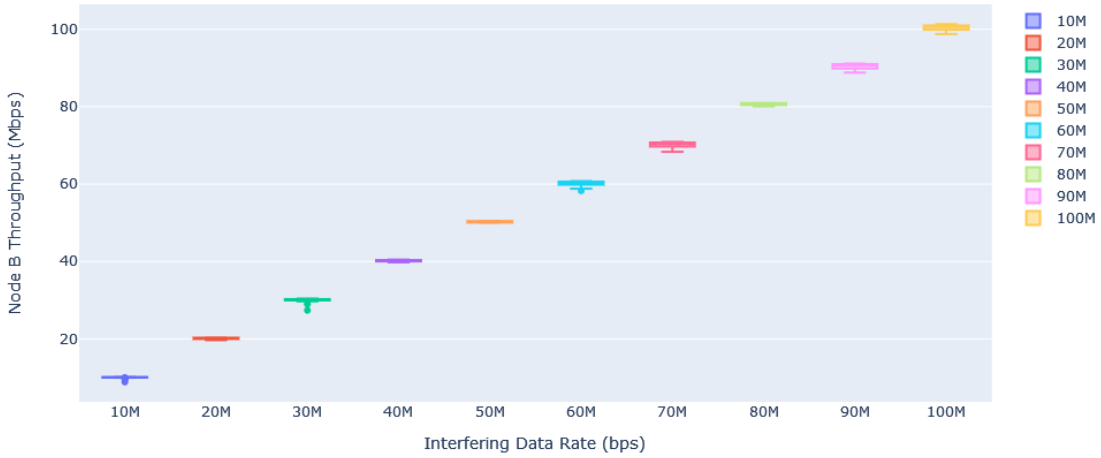


Figure 3.30: Node B Throughput with Interference 10-100 Mbps

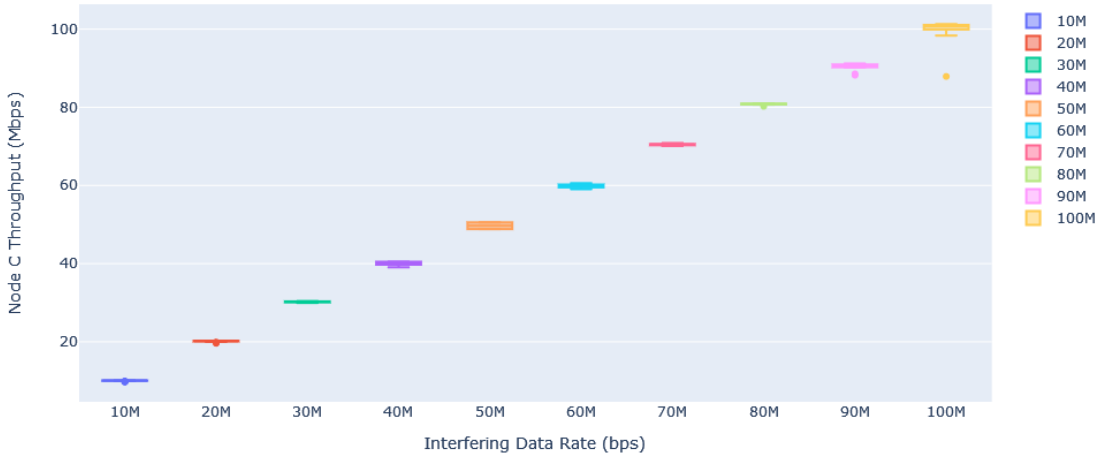


Figure 3.31: Node C Throughput with Interference 10-100 Mbps

3.4 Discussion

Through the developed work, we were able to analyze and observe how interference affects a Wi-Fi transmission.
Firstly a scenario without interference was simulated to obtain initial throughput values. In this scenario we were able to have a successful transmission with almost no lost packets. The throughput obtained is close to the maximum data rate that was set. If an initial higher data rate was configured for Node A, the throughput generated would be higher as well. Due to simulation issues that occurred during this work, obtaining such results proved rather difficult due to simulations freezing and not being able to obtain bigger sample sizes for certain simulations.

For small distance variations, such as in Section 3.3.2.3, there is no noticeable impact on obtained throughput. By putting an AP inside a small room, such as a Cath Lab, there should be no issues with a Wi-Fi transmission.

In Section 3.3.2.1 and Section 3.3.2.2 we can also conclude that within an ns-3 environment a longer simulation time can help reducing the variance of the measured throughput.

In Section 3.3.2.4 with the highest interference simulated of 100 Mbps per node totaling 200 Mbps of interfering data rate, the lowest achieved sample throughput was of 281 Mbps. With this throughput it is possible to send an uncompressed video of 480p at 30 fps with a color depth of 24 bits, since the necessary data rate for this video quality is given by:

$$DataRate = ColorDepth * XResolution * YResolution * FrameRate, \tag{3.4}$$

which is equal to 221 Mbps. By compressing the video using a video compression standard such as H.264, it is possible to transmit a picture of 1920x1080 resolution with a Frame Rate of 172 fps, or a 2560x1920 with 108 fps at 240 Mbps.

Unfortunately the ns-3 model only supports up to 4 spatial streams and 4 antennas. However, since the IEEE 802.11ax can use up to 8 spatial streams, it is theoretically possible to achieve double of the obtained throughput.

4

Conclusion

The purpose of this work was to evaluate on which conditions are possible to maintain a reliable constant bitrate of 400 Mbps on a IEEE 802.11ax network in the presence of interfering traffic. For this, the IEEE 802.11 standard was studied, providing an overview of WLAN technology. The IEEE 802.11 standard evolved to the current IEEE 802.11ax amendment, developed to tackle many problems, such as escalating demands for high throughput in densely populated scenarios, and it introduced notable enhancements in both the PHY and MAC layers. Through the ns-3 environment, we developed and computed multiple simulation scenarios depicting a Cath Lab, studying the influence of different parameters on the resulting throughput. We started with a simple simulation scenario going through all MCS value as well as GI and Channel Width. Afterward, we evaluated the difference in simulation duration and its impact on the results. Another simulation was executed that demonstrated the impact of the distance between communicating nodes and how it affects the obtained throughput. For small distances of up to 10 meters, no impact was noticed in obtained throughput. Finally, we simulated a scenario with a growing level of interference. The highest interference simulated was 200 Mbps total interfering data rate and the lowest throughput achieved was of 281 Mbps. With this throughput, it is possible to send an uncompressed video of 480p at 30 fps with a color depth of 24 bits. If we use video compression standards such as H.264, it is possible to transmit a picture of 1920x1080 at 172 fps, or a 2560x1920 at 108 fps. In conclusion, our exploration of diverse simulation scenarios within the IEEE 802.11ax framework has enriched our understanding of wireless communication in the demanding environment of a Cath Lab. The insights gained from our simulations pave the way for more robust, efficient, and reliable wireless networks in healthcare settings and beyond. As we conclude this phase of our research, we look forward to the continued evolution of WLAN technology and the myriad possibilities it holds for the future.

4.1 Future Work

The next step would be to simulate with other parameters that work better with high levels of interference such as a lower MCS, which is less vulnerable to noise. Other future possible simulations would include more BSSs and communicating nodes such as to simulate a more accurate hospital environment. Finally, the transmission power for a communication was not explored. Further simulations could be tested to see the effect of different transmission powers and how that could impact a scenario with interference as well as smaller distances and its impact can also be simulated.

Bibliography

[1] J. Bai et al. "Adaptive uplink OFDMA random access grouping scheme for ultra-dense networks in IEEE 802.11 ax". In: *2018 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE. 2018, pp. 34–39 (cit. on p. 16).

[2] O. Bejarano, E. W. Knightly, and M. Park. "IEEE 802.11 ac: from channelization to multi-user MIMO". In: *IEEE Communications Magazine* 51.10 (2013), pp. 84–90 (cit. on p. 12).

[3] B. Bellalta. "IEEE 802.11 ax: High-efficiency WLANs". In: *IEEE Wireless Communications* 23.1 (2016), pp. 38–46 (cit. on p. 5).

[4] B. Bellalta and K. Kosek-Szott. "AP-initiated multi-user transmissions in IEEE 802.11 ax WLANs". In: *Ad Hoc Networks* 85 (2019), pp. 145–159 (cit. on p. 10).

[5] CISCO. "Cisco Annual Internet Report (2018–2023)". In: (2020). URL: https://www.cisco.com/c/en/us/products/collateral/wireless/white-paper-c11-740788.pdf (cit. on p. 1).

[6] CISCO. "IEEE 802.11ax: The Sixth Generation of Wi-fi Technical white paper". In: (2018). URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf (cit. on p. 13).

[7] I. C. S. L. M. S. Committee et al. "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications". In: *ANSI/IEEE Std. 802.11-1999* (1999) (cit. on p. 3).

[8] Y. Daldoul, D.-E. Meddour, and A. Ksentini. "Performance evaluation of OFDMA and MU-MIMO in 802.11 ax networks". In: *Computer Networks* 182 (2020), p. 107477 (cit. on p. 12).

[9] D.-J. Deng et al. "IEEE 802.11 ax: highly efficient WLANs for intelligent information infrastructure". In: *IEEE Communications Magazine* 55.12 (2017), pp. 52–59 (cit. on p. 9).

[10] D.-J. Deng et al. "On quality-of-service provisioning in IEEE 802.11 ax WLANs". In: *IEEE Access* 4 (2016), pp. 6086–6104 (cit. on p. 17).

[11] Y. Feng et al. "A feasibility study of IEEE 802.11 HCCA for low-latency applications". In: *IEEE Transactions on Communications* 67.7 (2019), pp. 4928–4938 (cit. on p. 17).

[12] "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". In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), pp. 1–3534. DOI: 10.1109/IEEESTD.2016.7786995 (cit. on p. 12).

[13] E. Khorov et al. "A tutorial on IEEE 802.11 ax high efficiency WLANs". In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 197–216 (cit. on pp. 7, 16).

[14] H. Kim and J. So. "Improving Spatial Reuse of Wireless LAN Uplink Using BSS Color and Proximity Information". In: *Applied Sciences* 11.22 (2021), p. 11074 (cit. on p. 16).

[15] J. Lee. "OFDMA-based hybrid channel access for IEEE 802.11 ax WLAN". In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2018, pp. 188–193 (cit. on p. 17).

[16] T. Li, T. Tang, and C. Chang. "A new backoff algorithm for IEEE 802.11 distributed coordination function". In: *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 3. IEEE. 2009, pp. 455–459 (cit. on p. 14).

[17] J. M. Lourenço. *The NOVAthesis Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. i).

[18] D. Magrin et al. "Validation of the ns-3 802.11 ax OFDMA Implementation". In: *Proceedings of the 2021 Workshop on ns-3*. 2021, pp. 1–8 (cit. on p. 19).

[19] *MCS TABLE (UPDATED WITH 802.11AX DATA RATES)*. URL: https://semfionetworks.com/blog/mcs-table-updated-with-80211ax-data-rates/ (cit. on p. 11).

[20] S. Naribole, W. B. Lee, and A. Ranganath. "Impact of MU EDCA channel access on IEEE 802.11 ax WLANs". In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE. 2019, pp. 1–5 (cit. on p. 17).

[21] *Network Simulator ns-3 | a discrete-event network simulator for internet systems*. URL: https://www.nsnam.org/ (cit. on pp. 18, 20).

[22] *Network Simulator ns-3 | Model Library*. URL: https://www.nsnam.org/docs/release/3.34/models/html/index.html (cit. on p. 18).

[23] *Network Simulator ns-3 | Wifi-Design*. URL: https://www.nsnam.org/docs/release/3.34/models/html/wifi-design.html (cit. on p. 19).

[24] A. Networks. "White paper: 802.11ax". In: (2019). URL: https://www.arubanetworks.com/assets/wp/WP802.11AX.pdf (cit. on p. 7).

[25] S. Oh et al. "A novel call admission control scheme for the IEEE 802.11 e EDCA". In: *2008 10th International Conference on Advanced Communication Technology*. Vol. 3. IEEE. 2008, pp. 1832–1835 (cit. on p. 17).

[26] Q. Qu et al. "Survey and performance evaluation of the upcoming next generation WLANs standard-IEEE 802.11 ax". In: *Mobile Networks and Applications* 24.5 (2019), pp. 1461–1474 (cit. on p. 12).

[27] N. Šepić, E. Kočan, and M. Pejanović-Djurišić. "Evaluating spatial reuse in 802.11 ax networks with interference threshold adjustment". In: *2020 24th International Conference on Information Technology (IT)*. IEEE. 2020, pp. 1–4 (cit. on p. 15).

[28] N. Šepić et al. "Assessment of novel solutions for throughput enhancement in IEEE 802.11 ax networks". In: *2019 27th Telecommunications Forum (TELFOR)*. IEEE. 2019, pp. 1–4 (cit. on p. 16).

[29] G. L. Stuber et al. "Broadband MIMO-OFDM wireless communications". In: *Proceedings of the IEEE* 92.2 (2004), pp. 271–294 (cit. on pp. 5, 11).

[30] S. Sur et al. "Practical MU-MIMO user selection on 802.11 ac commodity networks". In: *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 2016, pp. 122–134 (cit. on p. 12).

[31] A. S. Tanenbaum, D. Wetherall, et al. *Computer networks*. 1996 (cit. on p. 6).

[32] F. Wilhelmi et al. "Spatial reuse in IEEE 802.11 ax WLANs". In: *arXiv preprint arXiv:1907.04141* (2019) (cit. on pp. 9, 15).

[33] H. I. Zawia, R. Hassan, and D. P. Dahnil. "A survey of medium access mechanisms for providing robust audio video streaming in IEEE 802.11 aa standard". In: *IEEE Access* 6 (2018), pp. 27690–27705 (cit. on p. 17).

A

Additional Generated Graphics

This chapter contains additional images generated throughout the simulations executed for this dissertation.

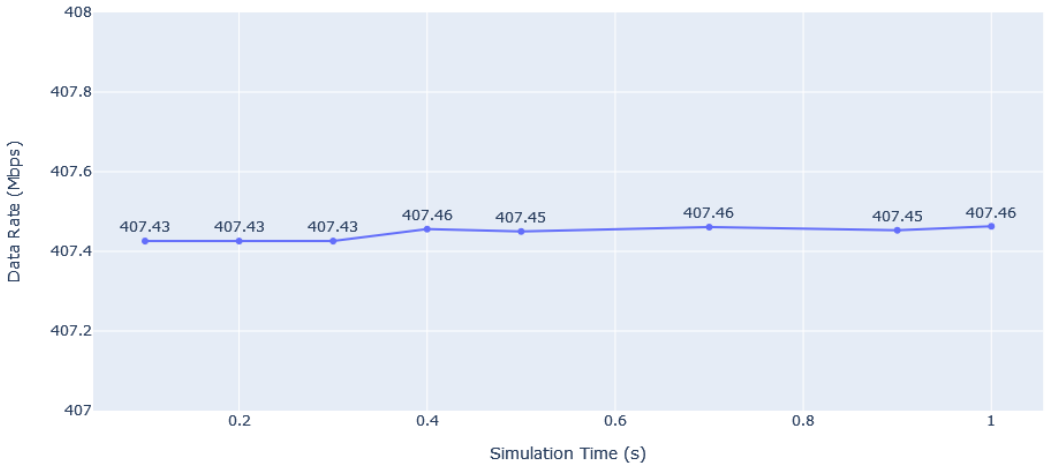


Figure A.1: Offered Data Rate across Simulation Time - Solo

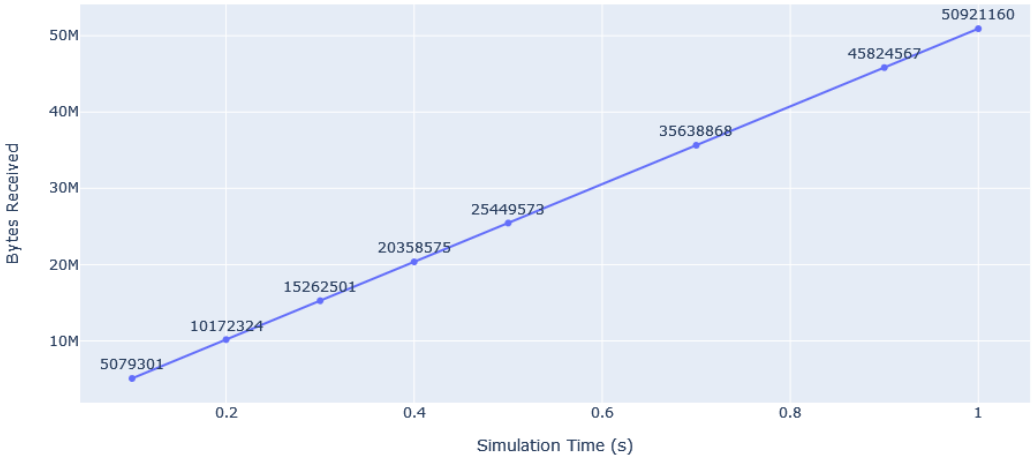


Figure A.2: Received Bytes across Simulation Time - Solo

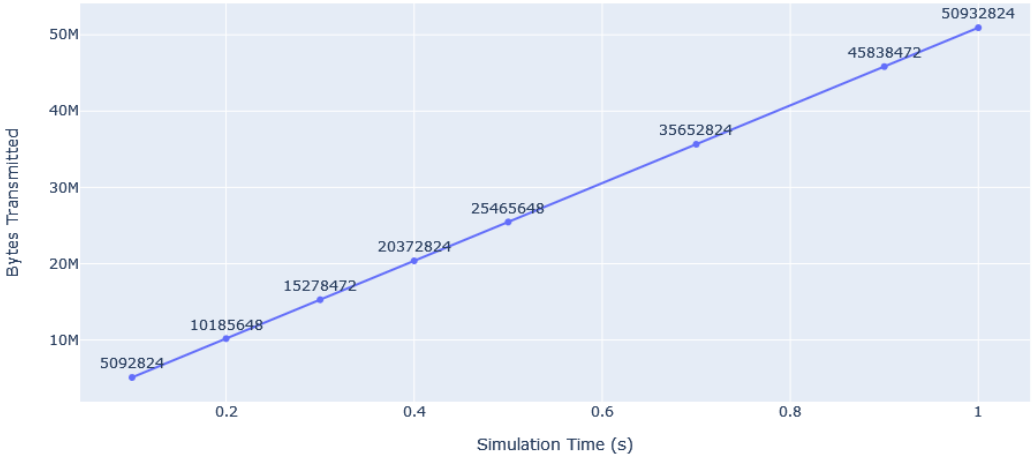


Figure A.3: Transmitted Bytes across Simulation Time - Solo

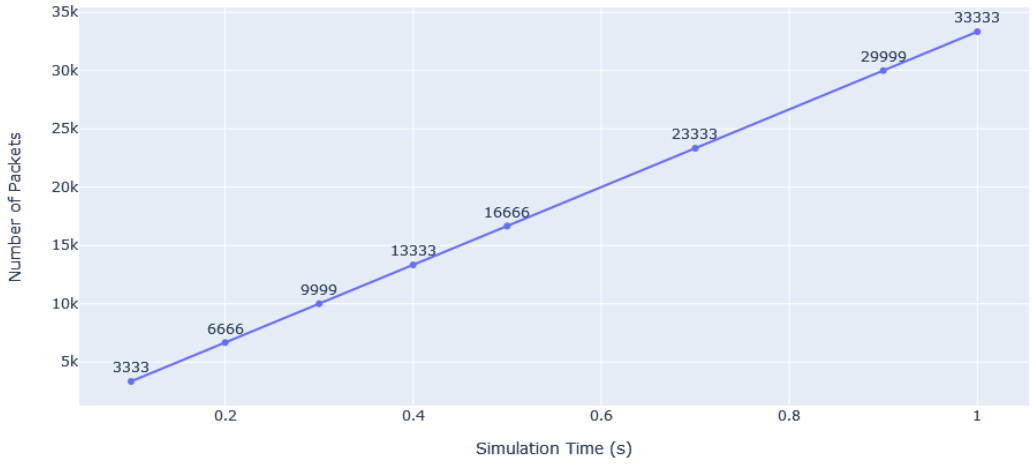


Figure A.4: Transmitted Packets across Simulation Time - Solo

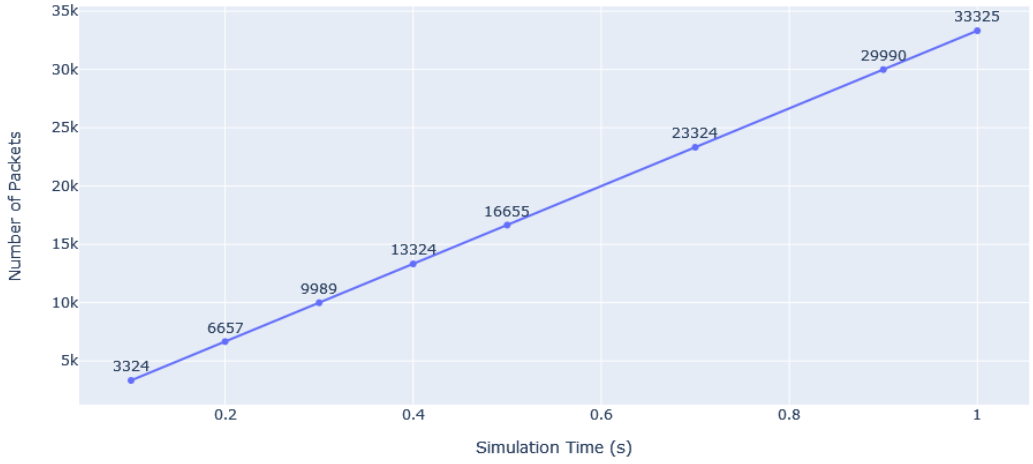


Figure A.5: Received Packets Over Simulation Time - Solo

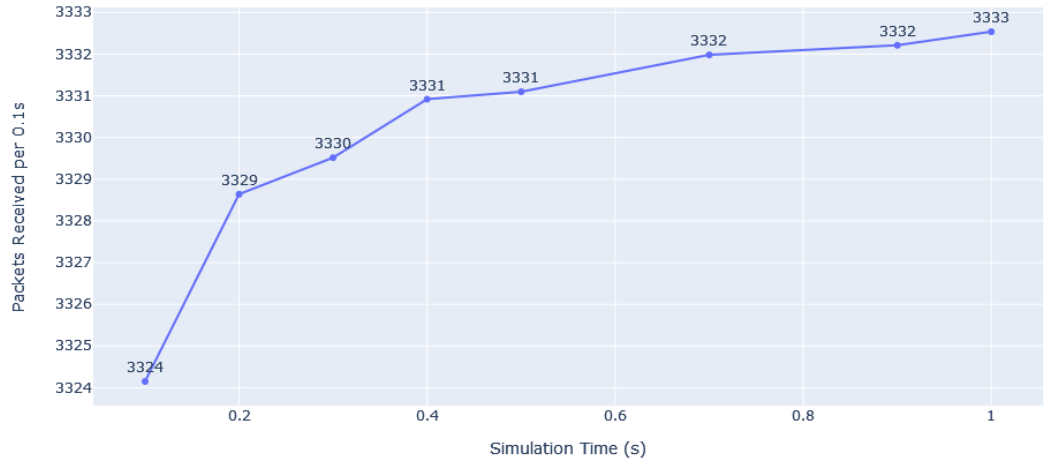


Figure A.6: Received Packets Per 0.1s of Simulation Time - Solo

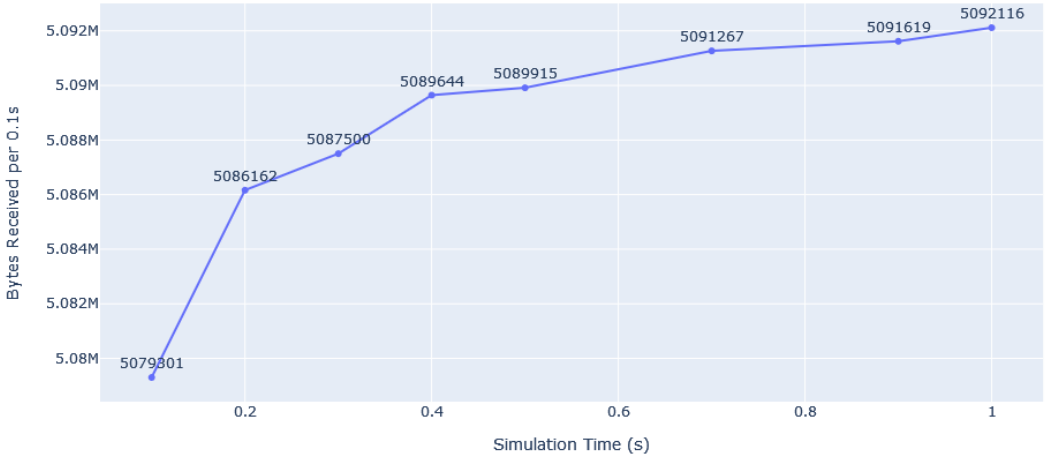


Figure A.7: Received Bytes Per 0.1s of Simulation Time - Solo

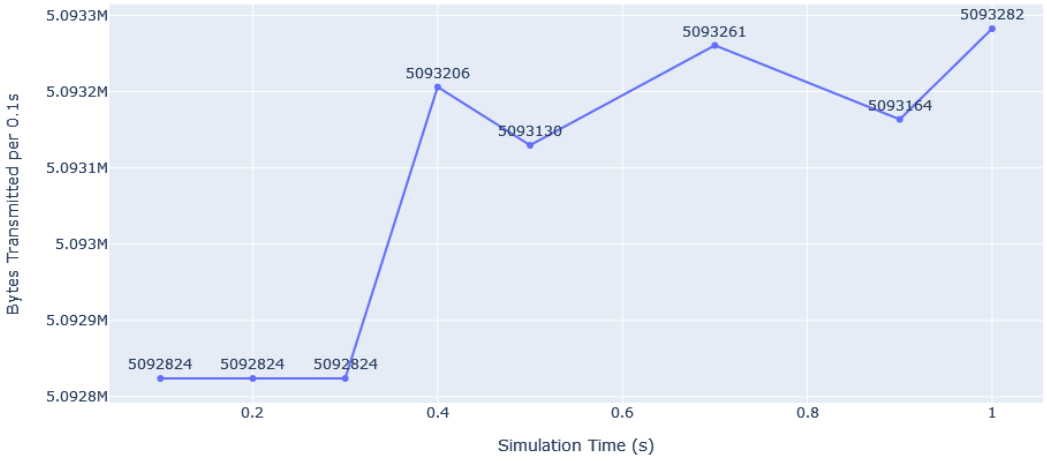


Figure A.8: Transmitted Bytes Per 0.1s of Simulation Time - Solo

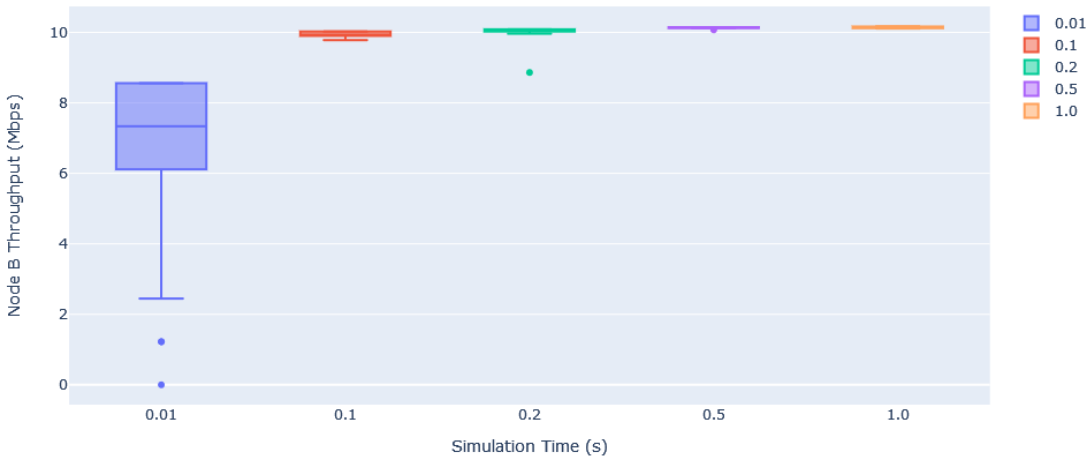


Figure A.9: Throughput over Simulation Time with 10Mbps of Interference - Node B

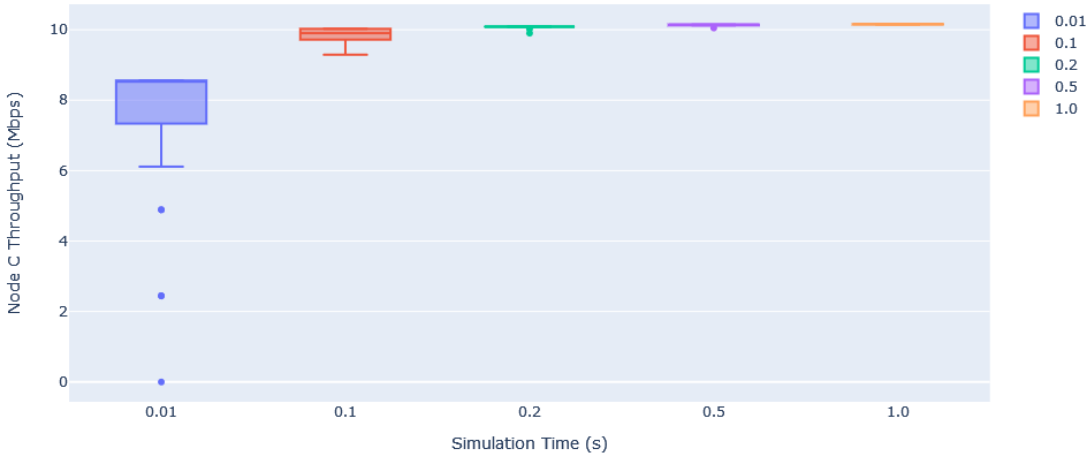


Figure A.10: Throughput over Simulation Time with 10Mbps of Interference - Node C

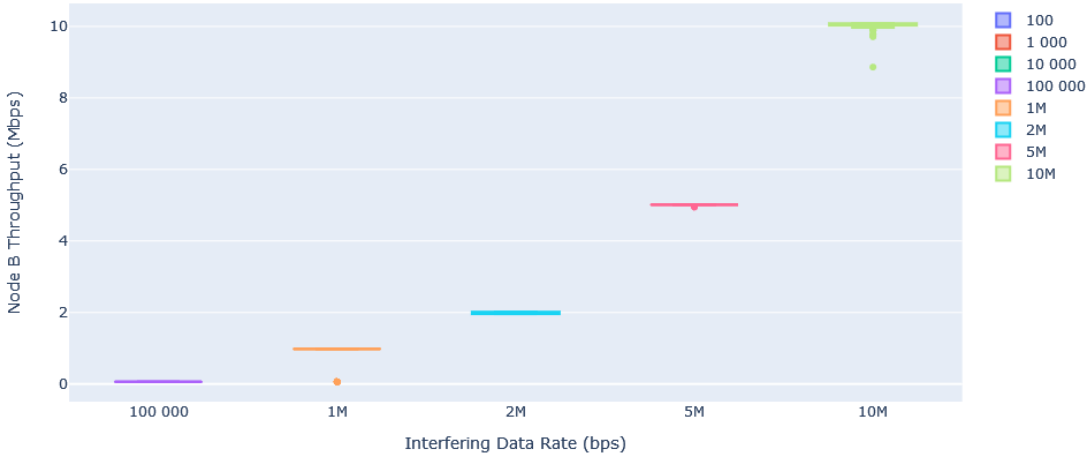


Figure A.11: Node B Throughput with Interference 1-10 Mbps

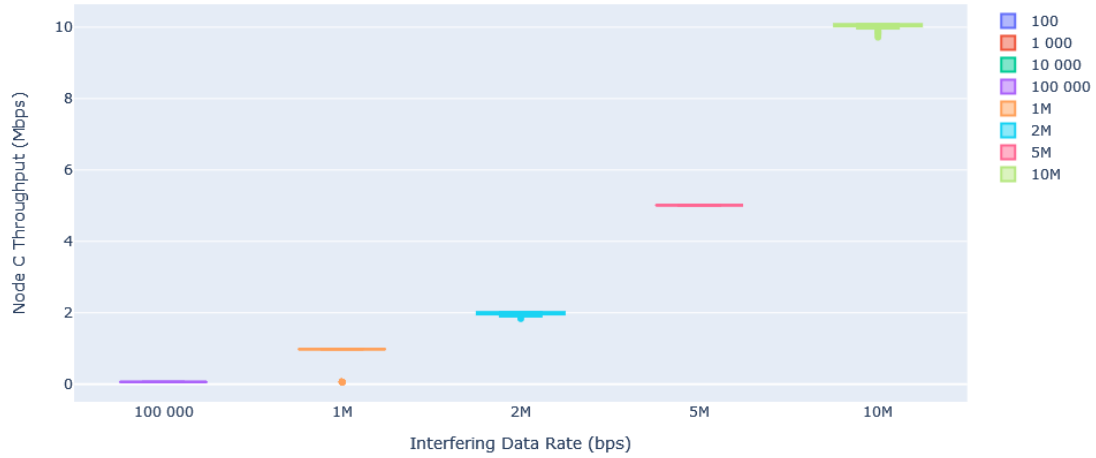


Figure A.12: Node C Throughput with Interference 1-10 Mbps

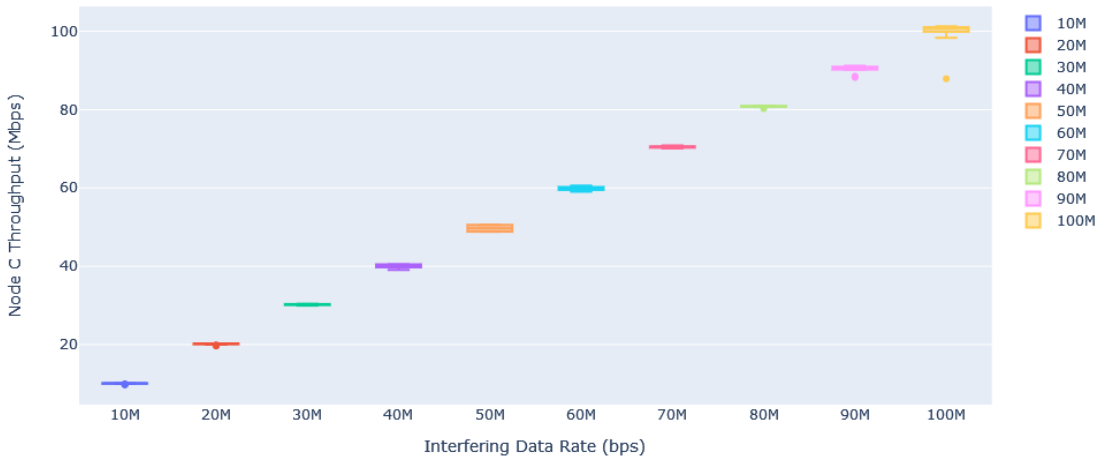


Figure A.13: Node C Throughput with Interference 10-100 Mbps

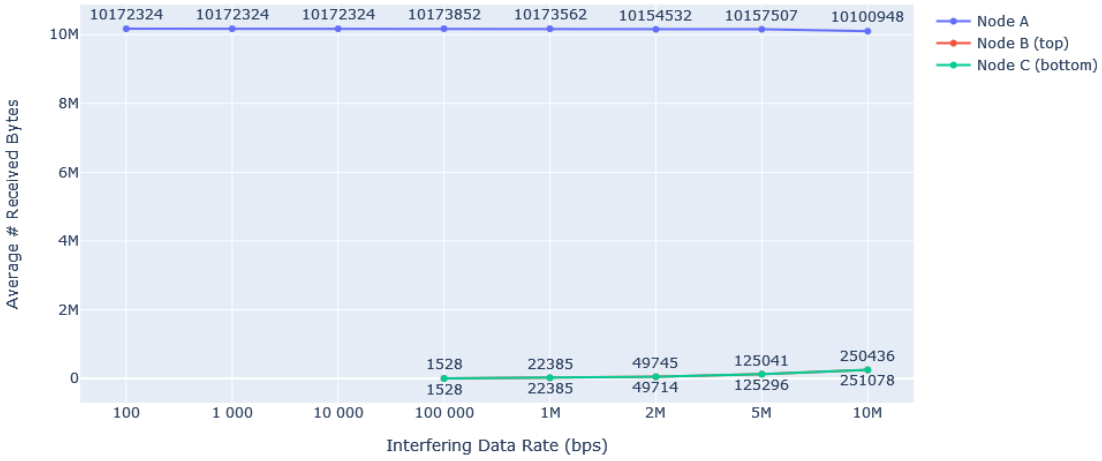


Figure A.14: Received Bytes with Interference 1-10 Mbps

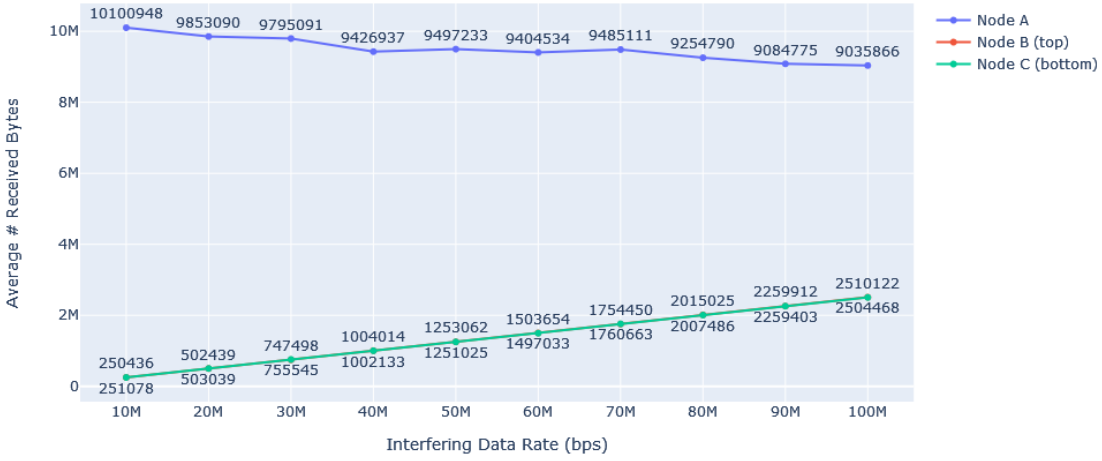


Figure A.15: Received Bytes with Interference 10-100 Mbps

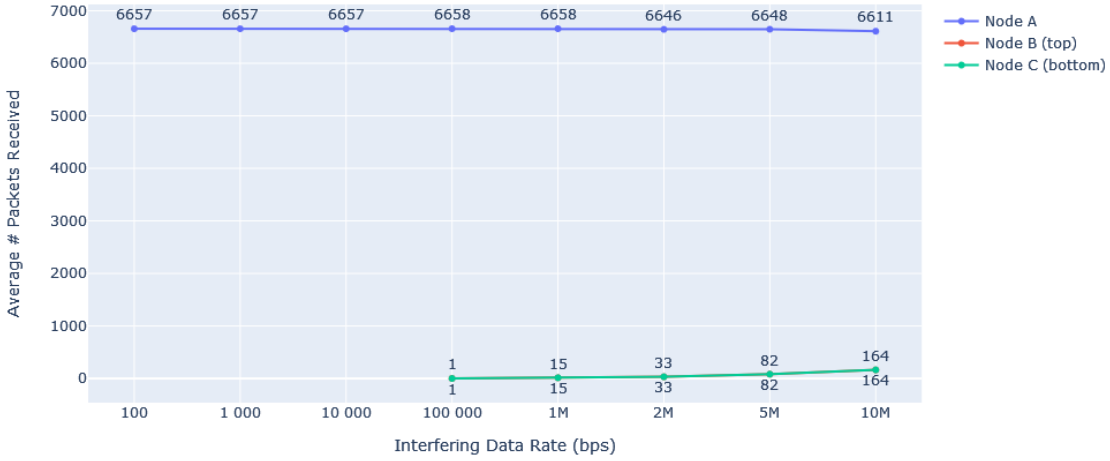


Figure A.16: Received Packets with Interference 1-10 Mbps

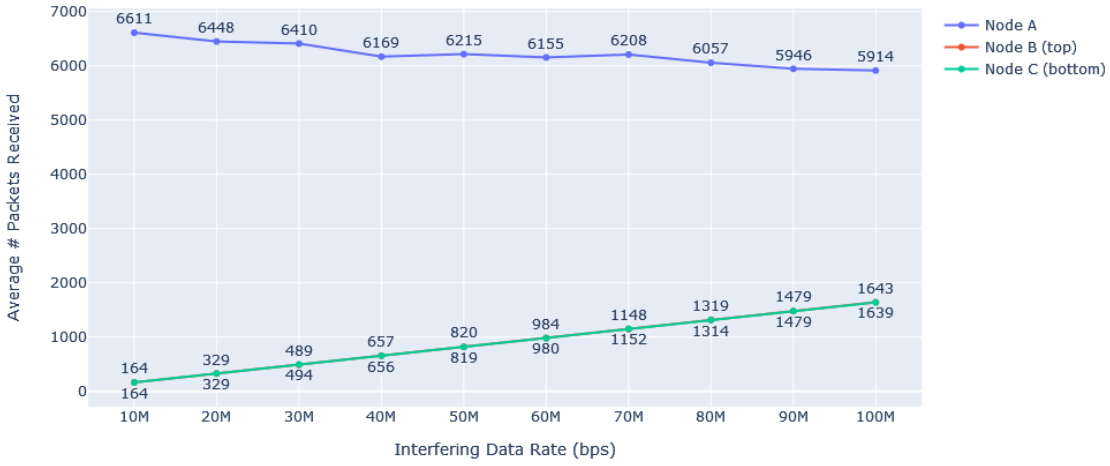


Figure A.17: Received Packets with Interference 10-100 Mbps

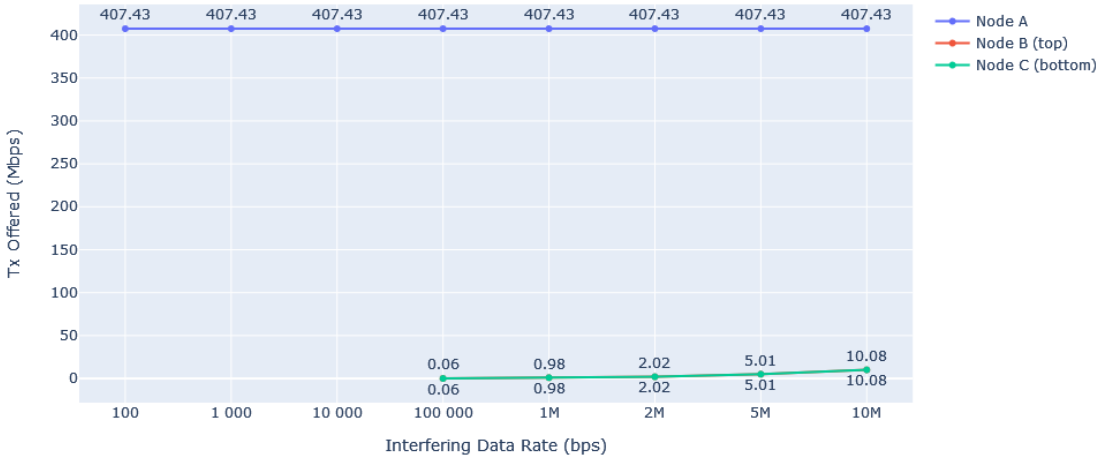


Figure A.18: Offered Data Rate with Interference 1-10 Mbps

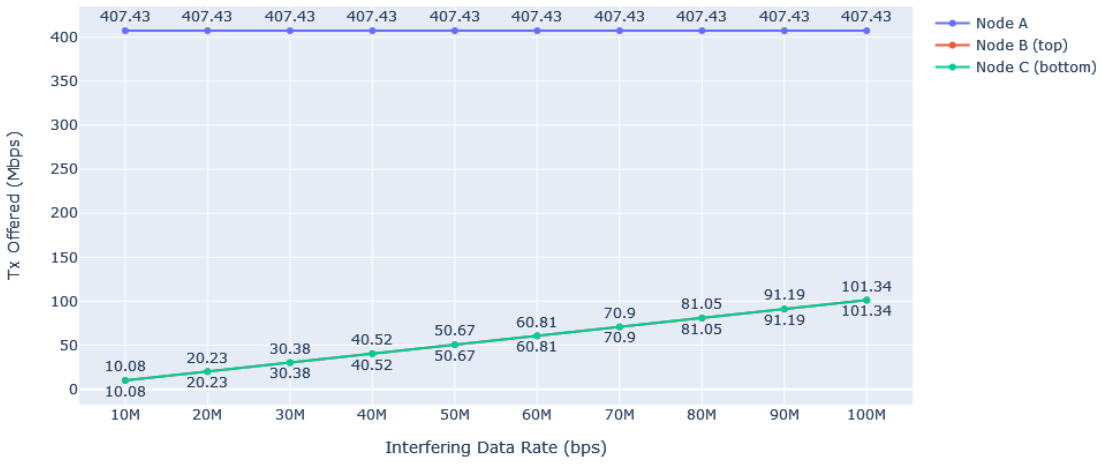


Figure A.19: Offered Data Rate with Interference 10-100 Mbps

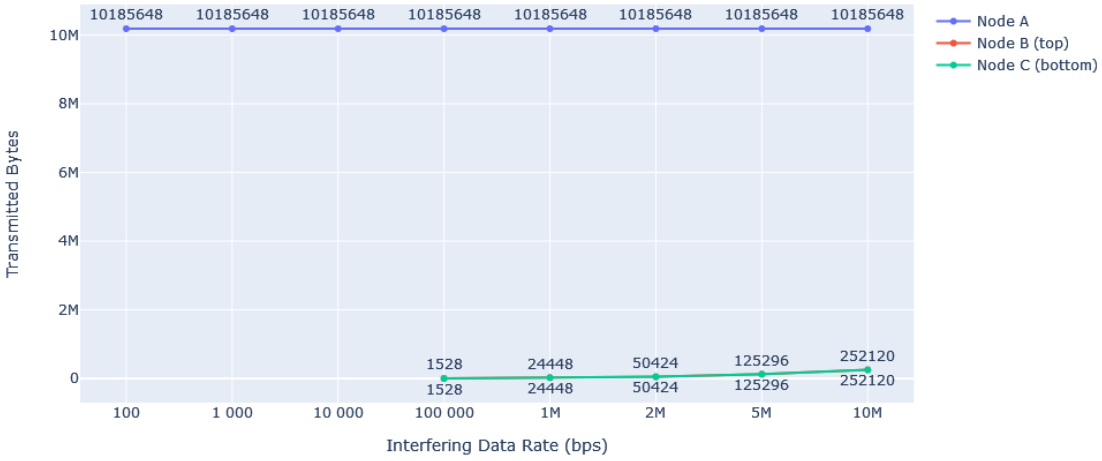


Figure A.20: Transmitted Bytes with Interference 1-10 Mbps

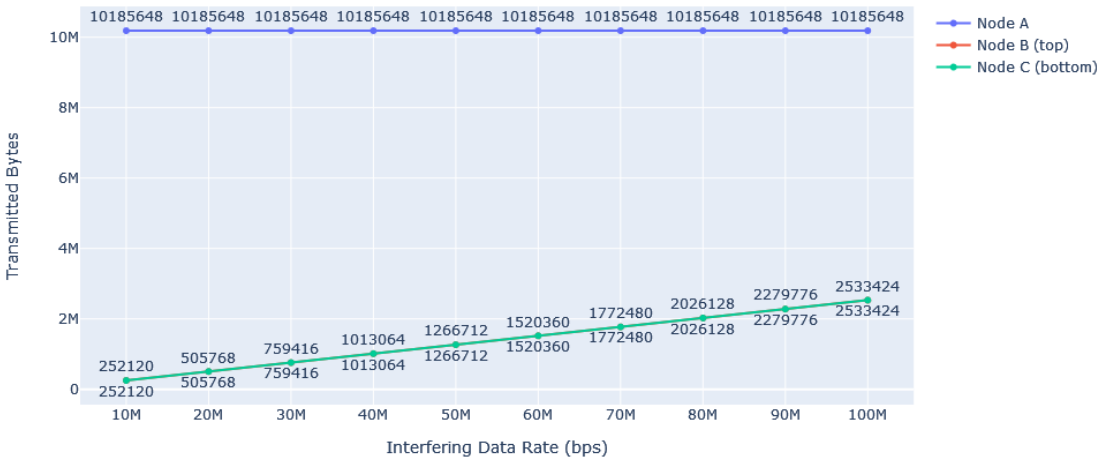


Figure A.21: Transmitted Bytes with Interference 10-100 Mbps

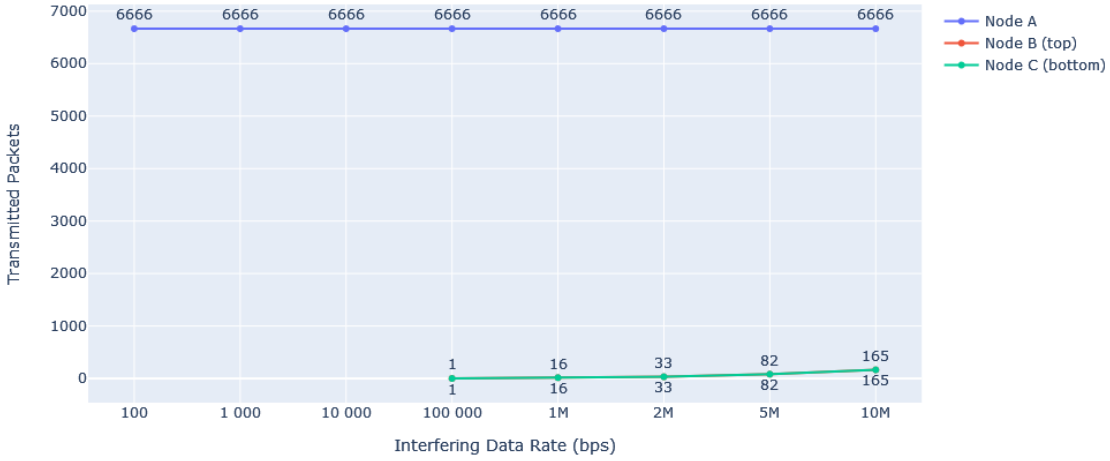


Figure A.22: Transmitted Packets with Interference 1-10 Mbps

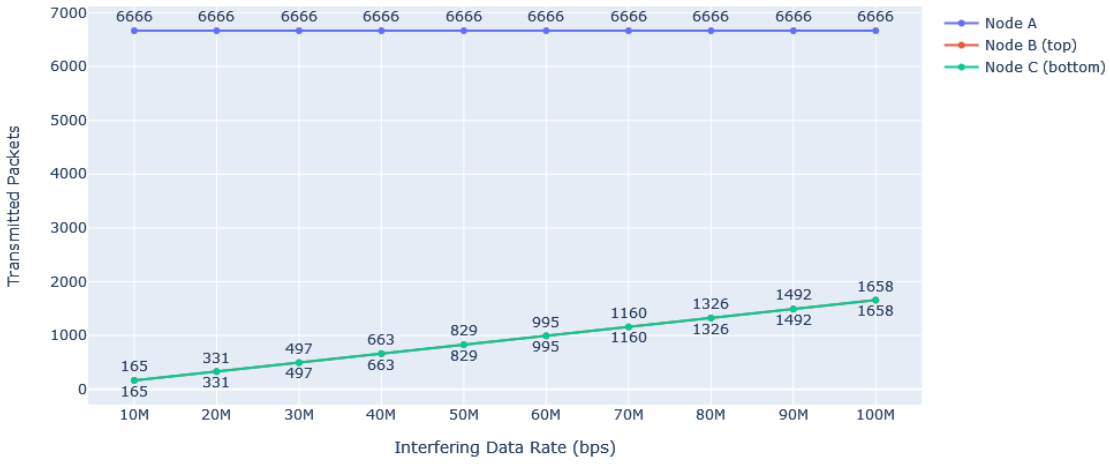


Figure A.23: Transmitted Packets with Interference 10-100 Mbps

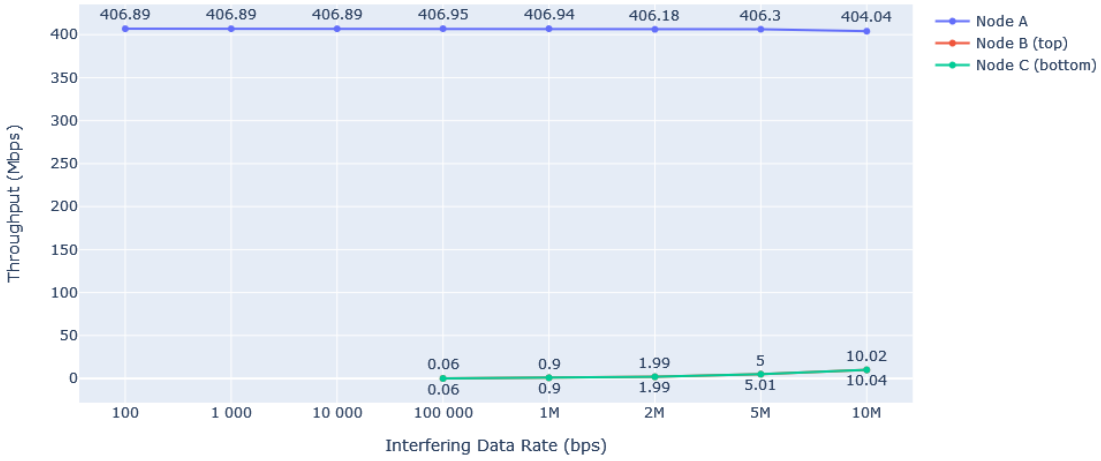


Figure A.24: Throughput with Interference 1-10 Mbps

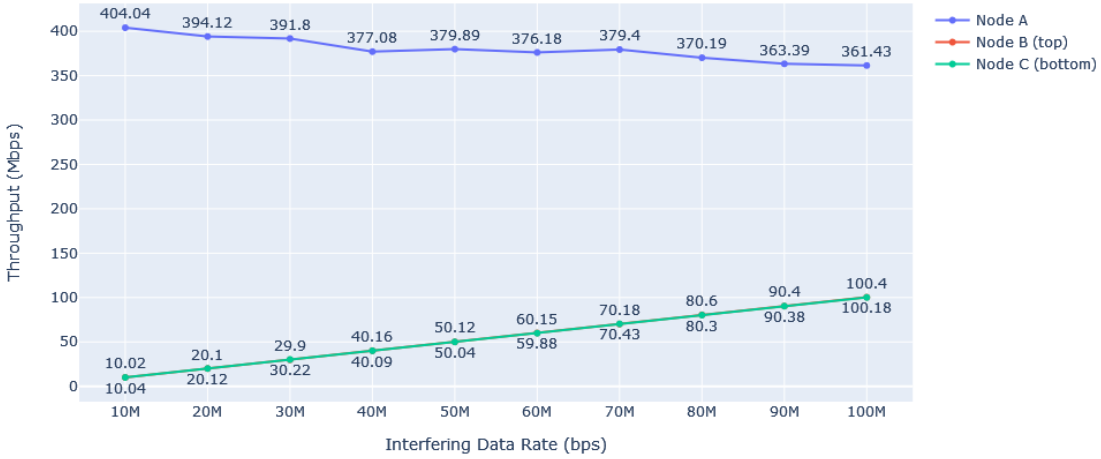


Figure A.25: Throughput with Interference 10-100 Mbps

B

Additional Generated Tables

This chapter contains the tables with all the generated values throughout the simulations executed for this dissertation.

Table B.1: Solo Simulation Values - 0.1s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	3333	5092824	407.426	3313	5062264	404.981
2	3333	5092824	407.426	3328	5085184	406.815
3	3333	5092824	407.426	3326	5082128	406.57
4	3333	5092824	407.426	3332	5091296	407.304
5	3333	5092824	407.426	3325	5080600	406.448
6	3333	5092824	407.426	3317	5068376	405.47
7	3333	5092824	407.426	3327	5083656	406.692
8	3333	5092824	407.426	3320	5072960	405.837
9	3333	5092824	407.426	3327	5083656	406.692
10	3333	5092824	407.426	3327	5083656	406.692
11	3333	5092824	407.426	3330	5088240	407.059
12	3333	5092824	407.426	3323	5077544	406.204
13	3333	5092824	407.426	3326	5082128	406.57
14	3333	5092824	407.426	3332	5091296	407.304
15	3333	5092824	407.426	3331	5089768	407.181
16	3333	5092824	407.426	3315	5065320	405.226
17	3333	5092824	407.426	3315	5065320	405.226
18	3333	5092824	407.426	3316	5066848	405.348
19	3333	5092824	407.426	3331	5089768	407.181
20	3333	5092824	407.426	3319	5071432	405.715
21	3333	5092824	407.426	3331	5089768	407.181
22	3333	5092824	407.426	3319	5071432	405.715
23	3333	5092824	407.426	3328	5085184	406.815
24	3333	5092824	407.426	3322	5076016	406.081
25	3333	5092824	407.426	3325	5080600	406.448
26	3333	5092824	407.426	3323	5077544	406.204
27	3333	5092824	407.426	3326	5082128	406.57
28	3333	5092824	407.426	3327	5083656	406.692
29	3333	5092824	407.426	3320	5072960	405.837
30	3333	5092824	407.426	3331	5089768	407.181
31	3333	5092824	407.426	3321	5074488	405.959
32	3333	5092824	407.426	3323	5077544	406.204
33	3333	5092824	407.426	3329	5086712	406.937
34	3333	5092824	407.426	3332	5091296	407.304
35	3333	5092824	407.426	3319	5071432	405.715
36	3333	5092824	407.426	3331	5089768	407.181
37	3333	5092824	407.426	3331	5089768	407.181
38	3333	5092824	407.426	3328	5085184	406.815
39	3333	5092824	407.426	3320	5072960	405.837
40	3333	5092824	407.426	3327	5083656	406.692
41	3333	5092824	407.426	3309	5056152	404.492
42	3333	5092824	407.426	3327	5083656	406.692
43	3333	5092824	407.426	3325	5080600	406.448
44	3333	5092824	407.426	3322	5076016	406.081
45	3333	5092824	407.426	3331	5089768	407.181
46	3333	5092824	407.426	3325	5080600	406.448
47	3333	5092824	407.426	3329	5086712	406.937
48	3333	5092824	407.426	3331	5089768	407.181
49	3333	5092824	407.426	3319	5071432	405.715
50	3333	5092824	407.426	3328	5085184	406.815
51	3333	5092824	407.426	3326	5082128	406.57
52	3333	5092824	407.426	3319	5071432	405.715
53	3333	5092824	407.426	3327	5083656	406.692
54	3333	5092824	407.426	3317	5068376	405.47
55	3333	5092824	407.426	3305	5050040	404.003
56	3333	5092824	407.426	3330	5088240	407.059
57	3333	5092824	407.426	3320	5072960	405.837
58	3333	5092824	407.426	3313	5062264	404.981
59	3333	5092824	407.426	3318	5069904	405.592
60	3333	5092824	407.426	3316	5066848	405.348
61	3333	5092824	407.426	3316	5066848	405.348
62	3333	5092824	407.426	3330	5088240	407.059
63	3333	5092824	407.426	3333	5092824	407.426
64	3333	5092824	407.426	3331	5089768	407.181
65	3333	5092824	407.426	3327	5083656	406.692
66	3333	5092824	407.426	3332	5091296	407.304
67	3333	5092824	407.426	3330	5088240	407.059
68	3333	5092824	407.426	3321	5074488	405.959
69	3333	5092824	407.426	3328	5085184	406.815
70	3333	5092824	407.426	3322	5076016	406.081
71	3333	5092824	407.426	3328	5085184	406.815
72	3333	5092824	407.426	3330	5088240	407.059
73	3333	5092824	407.426	3326	5082128	406.57
74	3333	5092824	407.426	3324	5079072	406.326
75	3333	5092824	407.426	3322	5076016	406.081
76	3333	5092824	407.426	3318	5069904	405.592
77	3333	5092824	407.426	3297	5037816	403.025
78	3333	5092824	407.426	3329	5086712	406.937
79	3333	5092824	407.426	3332	5091296	407.304
80	3333	5092824	407.426	3324	5079072	406.326
81	3333	5092824	407.426	3328	5085184	406.815
82	3333	5092824	407.426	3331	5089768	407.181
83	3333	5092824	407.426	3322	5076016	406.081
84	3333	5092824	407.426	3325	5080600	406.448
85	3333	5092824	407.426	3325	5080600	406.448
86	3333	5092824	407.426	3327	5083656	406.692
87	3333	5092824	407.426	3326	5082128	406.57
88	3333	5092824	407.426	3328	5085184	406.815
89	3333	5092824	407.426	3329	5086712	406.937
90	3333	5092824	407.426	3323	5077544	406.204
91	3333	5092824	407.426	3329	5086712	406.937
92	3333	5092824	407.426	3316	5066848	405.348
93	3333	5092824	407.426	3317	5068376	405.47
94	3333	5092824	407.426	3319	5071432	405.715
95	3333	5092824	407.426	3324	5079072	406.326
96	3333	5092824	407.426	3318	5069904	405.592
97	3333	5092824	407.426	3319	5071432	405.715
98	3333	5092824	407.426	3326	5082128	406.57
99	3333	5092824	407.426	3326	5082128	406.57
100	3333	5092824	407.426	3327	5083656	406.692

Table B.2: Solo Simulation Values - 0.2s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	6666	10185648	407.426	6656	10170368	406.815
2	6666	10185648	407.426	6661	10178008	407.12
3	6666	10185648	407.426	6652	10164256	406.57
4	6666	10185648	407.426	6656	10170368	406.815
5	6666	10185648	407.426	6650	10161200	406.448
6	6666	10185648	407.426	6665	10184120	407.365
7	6666	10185648	407.426	6649	10159672	406.387
8	6666	10185648	407.426	6661	10178008	407.12
9	6666	10185648	407.426	6665	10184120	407.365
10	6666	10185648	407.426	6661	10178008	407.12
11	6666	10185648	407.426	6658	10173424	406.937
12	6666	10185648	407.426	6659	10174952	406.998
13	6666	10185648	407.426	6618	10112304	404.492
14	6666	10185648	407.426	6665	10184120	407.365
15	6666	10185648	407.426	6662	10179536	407.181
16	6666	10185648	407.426	6656	10170368	406.815
17	6666	10185648	407.426	6655	10168840	406.754
18	6666	10185648	407.426	6665	10184120	407.365
19	6666	10185648	407.426	6654	10167312	406.692
20	6666	10185648	407.426	6650	10161200	406.448
21	6666	10185648	407.426	6648	10158144	406.326
22	6666	10185648	407.426	6650	10161200	406.448
23	6666	10185648	407.426	6616	10109248	404.37
24	6666	10185648	407.426	6662	10179536	407.181
25	6666	10185648	407.426	6664	10182592	407.304
26	6666	10185648	407.426	6653	10165784	406.631
27	6666	10185648	407.426	6662	10179536	407.181
28	6666	10185648	407.426	6661	10178008	407.12
29	6666	10185648	407.426	6649	10159672	406.387
30	6666	10185648	407.426	6662	10179536	407.181
31	6666	10185648	407.426	6660	10176480	407.059
32	6666	10185648	407.426	6660	10176480	407.059
33	6666	10185648	407.426	6662	10179536	407.181
34	6666	10185648	407.426	6655	10168840	406.754
35	6666	10185648	407.426	6660	10176480	407.059
36	6666	10185648	407.426	6662	10179536	407.181
37	6666	10185648	407.426	6662	10179536	407.181
38	6666	10185648	407.426	6660	10176480	407.059
39	6666	10185648	407.426	6660	10176480	407.059
40	6666	10185648	407.426	6652	10164256	406.57
41	6666	10185648	407.426	6663	10181064	407.243
42	6666	10185648	407.426	6660	10176480	407.059
43	6666	10185648	407.426	6664	10182592	407.304
44	6666	10185648	407.426	6661	10178008	407.12
45	6666	10185648	407.426	6666	10185648	407.426
46	6666	10185648	407.426	6663	10181064	407.243
47	6666	10185648	407.426	6647	10156616	406.265
48	6666	10185648	407.426	6654	10167312	406.692
49	6666	10185648	407.426	6659	10174952	406.998
50	6666	10185648	407.426	6662	10179536	407.181
51	6666	10185648	407.426	6651	10162728	406.509
52	6666	10185648	407.426	6661	10178008	407.12
53	6666	10185648	407.426	6629	10129112	405.164
54	6666	10185648	407.426	6659	10174952	406.998
55	6666	10185648	407.426	6661	10178008	407.12
56	6666	10185648	407.426	6660	10176480	407.059
57	6666	10185648	407.426	6662	10179536	407.181
58	6666	10185648	407.426	6657	10171896	406.876
59	6666	10185648	407.426	6650	10161200	406.448
60	6666	10185648	407.426	6660	10176480	407.059
61	6666	10185648	407.426	6660	10176480	407.059
62	6666	10185648	407.426	6656	10170368	406.815
63	6666	10185648	407.426	6651	10162728	406.509
64	6666	10185648	407.426	6649	10159672	406.387
65	6666	10185648	407.426	6662	10179536	407.181
66	6666	10185648	407.426	6661	10178008	407.12
67	6666	10185648	407.426	6663	10181064	407.243
68	6666	10185648	407.426	6657	10171896	406.876
69	6666	10185648	407.426	6663	10181064	407.243
70	6666	10185648	407.426	6656	10170368	406.815
71	6666	10185648	407.426	6665	10184120	407.365
72	6666	10185648	407.426	6651	10162728	406.509
73	6666	10185648	407.426	6656	10170368	406.815
74	6666	10185648	407.426	6659	10174952	406.998
75	6666	10185648	407.426	6650	10161200	406.448
76	6666	10185648	407.426	6656	10170368	406.815
77	6666	10185648	407.426	6657	10171896	406.876
78	6666	10185648	407.426	6664	10182592	407.304
79	6666	10185648	407.426	6661	10178008	407.12
80	6666	10185648	407.426	6663	10181064	407.243
81	6666	10185648	407.426	6659	10174952	406.998
82	6666	10185648	407.426	6653	10165784	406.631
83	6666	10185648	407.426	6662	10179536	407.181
84	6666	10185648	407.426	6658	10173424	406.937
85	6666	10185648	407.426	6666	10185648	407.426
86	6666	10185648	407.426	6653	10165784	406.631
87	6666	10185648	407.426	6663	10181064	407.243
88	6666	10185648	407.426	6658	10173424	406.937
89	6666	10185648	407.426	6666	10185648	407.426
90	6666	10185648	407.426	6658	10173424	406.937
91	6666	10185648	407.426	6661	10178008	407.12
92	6666	10185648	407.426	6666	10185648	407.426
93	6666	10185648	407.426	6661	10178008	407.12
94	6666	10185648	407.426	6658	10173424	406.937
95	6666	10185648	407.426	6650	10161200	406.448
96	6666	10185648	407.426	6652	10164256	406.57
97	6666	10185648	407.426	6651	10162728	406.509
98	6666	10185648	407.426	6663	10181064	407.243
99	6666	10185648	407.426	6660	10176480	407.059
100	6666	10185648	407.426	6663	10181064	407.243

Table B.3: Solo Simulation Values - 0.3s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	9999	15278472	407.426	9996	15273888	407.304
2	9999	15278472	407.426	9983	15254024	406.774
3	9999	15278472	407.426	9993	15269304	407.181
4	9999	15278472	407.426	9991	15266248	407.1
5	9999	15278472	407.426	9997	15275416	407.344
6	9999	15278472	407.426	9993	15269304	407.181
7	9999	15278472	407.426	9997	15275416	407.344
8	9999	15278472	407.426	9991	15266248	407.1
9	9999	15278472	407.426	9988	15261664	406.978
10	9999	15278472	407.426	9995	15272360	407.263
11	9999	15278472	407.426	9941	15189848	405.063
12	9999	15278472	407.426	9985	15257080	406.855
13	9999	15278472	407.426	9994	15270832	407.222
14	9999	15278472	407.426	9989	15263192	407.018
15	9999	15278472	407.426	9998	15276944	407.385
16	9999	15278472	407.426	9986	15258608	406.896
17	9999	15278472	407.426	9993	15269304	407.181
18	9999	15278472	407.426	9955	15211240	405.633
19	9999	15278472	407.426	9987	15260136	406.937
20	9999	15278472	407.426	9999	15278472	407.426
21	9999	15278472	407.426	9980	15249440	406.652
22	9999	15278472	407.426	9992	15267776	407.141
23	9999	15278472	407.426	9993	15269304	407.181
24	9999	15278472	407.426	9984	15255552	406.815
25	9999	15278472	407.426	9998	15276944	407.385
26	9999	15278472	407.426	9994	15270832	407.222
27	9999	15278472	407.426	9983	15254024	406.774
28	9999	15278472	407.426	9992	15267776	407.141
29	9999	15278472	407.426	9984	15255552	406.815
30	9999	15278472	407.426	9978	15246384	406.57
31	9999	15278472	407.426	9991	15266248	407.1
32	9999	15278472	407.426	9998	15276944	407.385
33	9999	15278472	407.426	9994	15270832	407.222
34	9999	15278472	407.426	9998	15276944	407.385
35	9999	15278472	407.426	9995	15272360	407.263
36	9999	15278472	407.426	9989	15263192	407.018
37	9999	15278472	407.426	9989	15263192	407.018
38	9999	15278472	407.426	9992	15267776	407.141
39	9999	15278472	407.426	9990	15264720	407.059
40	9999	15278472	407.426	9986	15258608	406.896
41	9999	15278472	407.426	9969	15232632	406.204
42	9999	15278472	407.426	9999	15278472	407.426

Table B.4: Solo Simulation Values - 0.4s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	13333	20372824	407.456	13316	20346848	406.937
2	13333	20372824	407.456	13327	20363656	407.273
3	13333	20372824	407.456	13329	20366712	407.334
4	13333	20372824	407.456	13332	20371296	407.426
5	13333	20372824	407.456	13333	20372824	407.456
6	13333	20372824	407.456	13317	20348376	406.968
7	13333	20372824	407.456	13328	20365184	407.304
8	13333	20372824	407.456	13319	20351432	407.029
9	13333	20372824	407.456	13327	20363656	407.273
10	13333	20372824	407.456	13333	20372824	407.456
11	13333	20372824	407.456	13331	20369768	407.395
12	13333	20372824	407.456	13328	20365184	407.304
13	13333	20372824	407.456	13328	20365184	407.304
14	13333	20372824	407.456	13329	20366712	407.334
15	13333	20372824	407.456	13330	20368240	407.365
16	13333	20372824	407.456	13321	20354488	407.09
17	13333	20372824	407.456	13327	20363656	407.273
18	13333	20372824	407.456	13320	20352960	407.059
19	13333	20372824	407.456	13329	20366712	407.334
20	13333	20372824	407.456	13320	20352960	407.059
21	13333	20372824	407.456	13327	20363656	407.273
22	13333	20372824	407.456	13332	20371296	407.426
23	13333	20372824	407.456	13293	20311704	406.234
24	13333	20372824	407.456	13321	20354488	407.09
25	13333	20372824	407.456	13328	20365184	407.304
26	13333	20372824	407.456	13328	20365184	407.304
27	13333	20372824	407.456	13325	20360600	407.212
28	13333	20372824	407.456	13327	20363656	407.273
29	13333	20372824	407.456	13333	20372824	407.456
30	13333	20372824	407.456	13327	20363656	407.273
31	13333	20372824	407.456	13332	20371296	407.426
32	13333	20372824	407.456	13310	20337680	406.754
33	13333	20372824	407.456	13313	20342264	406.845
34	13333	20372824	407.456	13327	20363656	407.273
35	13333	20372824	407.456	13306	20331568	406.631
36	13333	20372824	407.456	13315	20345320	406.906
37	13333	20372824	407.456	13312	20340736	406.815
38	13333	20372824	407.456	13320	20352960	407.059
39	13333	20372824	407.456	13320	20352960	407.059
40	13333	20372824	407.456	13327	20363656	407.273

Table B.5: Solo Simulation Values - 0.5s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	16666	25465648	407.45	16663	25461064	407.377
2	16666	25465648	407.45	16653	25445784	407.133
3	16666	25465648	407.45	16659	25454952	407.279
4	16666	25465648	407.45	16662	25459536	407.353
5	16666	25465648	407.45	16660	25456480	407.304
6	16666	25465648	407.45	16662	25459536	407.353
7	16666	25465648	407.45	16664	25462592	407.401
8	16666	25465648	407.45	16654	25447312	407.157
9	16666	25465648	407.45	16611	25381608	406.106
10	16666	25465648	407.45	16658	25453424	407.255
11	16666	25465648	407.45	16663	25461064	407.377
12	16666	25465648	407.45	16658	25453424	407.255
13	16666	25465648	407.45	16655	25448840	407.181
14	16666	25465648	407.45	16654	25447312	407.157
15	16666	25465648	407.45	16660	25456480	407.304
16	16666	25465648	407.45	16653	25445784	407.133
17	16666	25465648	407.45	16651	25442728	407.084
18	16666	25465648	407.45	16644	25432032	406.913
19	16666	25465648	407.45	16655	25448840	407.181
20	16666	25465648	407.45	16665	25464120	407.426
21	16666	25465648	407.45	16663	25461064	407.377
22	16666	25465648	407.45	16652	25444256	407.108
23	16666	25465648	407.45	16660	25456480	407.304
24	16666	25465648	407.45	16656	25450368	407.206
25	16666	25465648	407.45	16652	25444256	407.108

Table B.6: Solo Simulation Values - 0.7s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	23333	35652824	407.461	23324	35639072	407.304
2	23333	35652824	407.461	23330	35648240	407.408
3	23333	35652824	407.461	23320	35632960	407.234
4	23333	35652824	407.461	23317	35628376	407.181
5	23333	35652824	407.461	23316	35626848	407.164
6	23333	35652824	407.461	23326	35642128	407.339
7	23333	35652824	407.461	23331	35649768	407.426
8	23333	35652824	407.461	23301	35603928	406.902
9	23333	35652824	407.461	23321	35634488	407.251
10	23333	35652824	407.461	23326	35642128	407.339
11	23333	35652824	407.461	23316	35626848	407.164
12	23333	35652824	407.461	23326	35642128	407.339
13	23333	35652824	407.461	23300	35602400	406.885
14	23333	35652824	407.461	23332	35651296	407.443
15	23333	35652824	407.461	23332	35651296	407.443
16	23333	35652824	407.461	23323	35637544	407.286
17	23333	35652824	407.461	23331	35649768	407.426
18	23333	35652824	407.461	23327	35643656	407.356
19	23333	35652824	407.461	23320	35632960	407.234
20	23333	35652824	407.461	23324	35639072	407.304
21	23333	35652824	407.461	23333	35652824	407.461
22	23333	35652824	407.461	23331	35649768	407.426
23	23333	35652824	407.461	23328	35645184	407.374
24	23333	35652824	407.461	23327	35643656	407.356
25	23333	35652824	407.461	23328	35645184	407.374
26	23333	35652824	407.461	23318	35629904	407.199
27	23333	35652824	407.461	23319	35631432	407.216
28	23333	35652824	407.461	23310	35617680	407.059
29	23333	35652824	407.461	23328	35645184	407.374
30	23333	35652824	407.461	23326	35642128	407.339
31	23333	35652824	407.461	23327	35643656	407.356
32	23333	35652824	407.461	23328	35645184	407.374
33	23333	35652824	407.461	23330	35648240	407.408
34	23333	35652824	407.461	23330	35648240	407.408
35	23333	35652824	407.461	23327	35643656	407.356
36	23333	35652824	407.461	23325	35640600	407.321
37	23333	35652824	407.461	23331	35649768	407.426
38	23333	35652824	407.461	23305	35610040	406.972
39	23333	35652824	407.461	23327	35643656	407.356
40	23333	35652824	407.461	23331	35649768	407.426
41	23333	35652824	407.461	23331	35649768	407.426
42	23333	35652824	407.461	23326	35642128	407.339
43	23333	35652824	407.461	23316	35626848	407.164
44	23333	35652824	407.461	23331	35649768	407.426
45	23333	35652824	407.461	23328	35645184	407.374
46	23333	35652824	407.461	23327	35643656	407.356
47	23333	35652824	407.461	23322	35636016	407.269
48	23333	35652824	407.461	23320	35632960	407.234
49	23333	35652824	407.461	23320	35632960	407.234
50	23333	35652824	407.461	23329	35646712	407.391
51	23333	35652824	407.461	23325	35640600	407.321
52	23333	35652824	407.461	23303	35606984	406.937
53	23333	35652824	407.461	23328	35645184	407.374
54	23333	35652824	407.461	23325	35640600	407.321
55	23333	35652824	407.461	23322	35636016	407.269
56	23333	35652824	407.461	23332	35651296	407.443
57	23333	35652824	407.461	23330	35648240	407.408
58	23333	35652824	407.461	23321	35634488	407.251
59	23333	35652824	407.461	23332	35651296	407.443
60	23333	35652824	407.461	23312	35620736	407.094

Table B.7: Solo Simulation Values - 0.9s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	29999	45838472	407.453	29995	45832360	407.399
2	29999	45838472	407.453	29999	45838472	407.453
3	29999	45838472	407.453	29982	45812496	407.222
4	29999	45838472	407.453	29990	45824720	407.331
5	29999	45838472	407.453	29986	45818608	407.277
6	29999	45838472	407.453	29988	45821664	407.304
7	29999	45838472	407.453	29997	45835416	407.426
8	29999	45838472	407.453	29997	45835416	407.426
9	29999	45838472	407.453	29989	45823192	407.317
10	29999	45838472	407.453	29998	45836944	407.44
11	29999	45838472	407.453	29985	45817080	407.263
12	29999	45838472	407.453	29986	45818608	407.277
13	29999	45838472	407.453	29977	45804856	407.154
14	29999	45838472	407.453	29983	45814024	407.236
15	29999	45838472	407.453	29985	45817080	407.263
16	29999	45838472	407.453	29998	45836944	407.44
17	29999	45838472	407.453	29992	45827776	407.358
18	29999	45838472	407.453	29993	45829304	407.372
19	29999	45838472	407.453	29987	45820136	407.29
20	29999	45838472	407.453	29991	45826248	407.344

Table B.8: Solo Simulation Values - 1.0s

RNG	Tx Packets	Tx Bytes	Tx Offered	Rx Packets	Rx Bytes	Throughput
1	33333	50932824	407.463	33329	50926712	407.414
2	33333	50932824	407.463	33331	50929768	407.438
3	33333	50932824	407.463	33328	50925184	407.401
4	33333	50932824	407.463	33319	50911432	407.291
5	33333	50932824	407.463	33327	50923656	407.389
6	33333	50932824	407.463	33325	50920600	407.365
7	33333	50932824	407.463	33323	50917544	407.34
8	33333	50932824	407.463	33331	50929768	407.438
9	33333	50932824	407.463	33331	50929768	407.438
10	33333	50932824	407.463	33322	50916016	407.328
11	33333	50932824	407.463	33314	50903792	407.23
12	33333	50932824	407.463	33327	50923656	407.389
13	33333	50932824	407.463	33331	50929768	407.438
14	33333	50932824	407.463	33319	50911432	407.291
15	33333	50932824	407.463	33328	50925184	407.401
16	33333	50932824	407.463	33320	50912960	407.304
17	33333	50932824	407.463	33318	50909904	407.279
18	33333	50932824	407.463	33332	50931296	407.45
19	33333	50932824	407.463	33328	50925184	407.401
20	33333	50932824	407.463	33329	50926712	407.414
21	33333	50932824	407.463	33324	50919072	407.353
22	33333	50932824	407.463	33322	50916016	407.328
23	33333	50932824	407.463	33331	50929768	407.438
24	33333	50932824	407.463	33320	50912960	407.304
25	33333	50932824	407.463	33327	50923656	407.389
26	33333	50932824	407.463	33332	50931296	407.45
27	33333	50932824	407.463	33322	50916016	407.328
28	33333	50932824	407.463	33319	50911432	407.291
29	33333	50932824	407.463	33324	50919072	407.353
30	33333	50932824	407.463	33328	50925184	407.401

Table B.9: Interference 0.01 second

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	333	508824	407.059	276	421728	337.382	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
2	333	508824	407.059	183	279624	223.699	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
3	333	508824	407.059	265	404920	323.936	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
4	333	508824	407.059	329	502712	402.17	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
5	333	508824	407.059	233	356024	284.819	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
6	333	508824	407.059	255	389640	311.712	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
7	333	508824	407.059	314	479792	383.834	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
8	333	508824	407.059	206	314768	251.814	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
9	333	508824	407.059	326	498128	398.502	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
10	333	508824	407.059	259	395752	316.602	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
11	333	508824	407.059	304	464512	371.61	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
12	333	508824	407.059	295	450760	360.608	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
13	333	508824	407.059	290	443120	354.496	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
14	333	508824	407.059	326	498128	398.502	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	2	3056	2.4448
15	333	508824	407.059	221	337688	270.15	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
16	333	508824	407.059	321	490488	392.39	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
17	333	508824	407.059	274	418672	334.938	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
18	333	508824	407.059	324	495072	396.058	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
19	333	508824	407.059	312	476736	381.389	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
20	333	508824	407.059	193	294904	235.923	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
21	333	508824	407.059	267	407976	326.381	7	10696	8.5568	0	0	0	7	10696	8.5568	7	10696	8.5568
22	333	508824	407.059	223	340744	272.595	7	10696	8.5568	3	4584	3.6672	7	10696	8.5568	7	10696	8.5568
23	333	508824	407.059	208	317824	254.259	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
24	333	508824	407.059	318	485904	388.723	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
25	333	508824	407.059	311	475208	380.166	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
26	333	508824	407.059	327	499656	399.725	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
27	333	508824	407.059	299	456872	365.498	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
28	333	508824	407.059	266	406448	325.158	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
29	333	508824	407.059	328	501184	400.947	7	10696	8.5568	3	4584	3.6672	7	10696	8.5568	5	7640	6.112
30	333	508824	407.059	301	459928	367.942	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
31	333	508824	407.059	218	333104	266.483	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
32	333	508824	407.059	290	443120	354.496	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	4	6112	4.8896
33	333	508824	407.059	281	429368	343.494	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
34	333	508824	407.059	279	426312	341.05	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
35	333	508824	407.059	283	432424	345.939	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
36	333	508824	407.059	235	359080	287.264	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
37	333	508824	407.059	302	461456	369.165	7	10696	8.5568	4	6112	4.8896	7	10696	8.5568	7	10696	8.5568
38	333	508824	407.059	309	472152	377.722	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
39	333	508824	407.059	287	438536	350.829	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
40	333	508824	407.059	216	330048	264.038	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
41	333	508824	407.059	200	305600	244.48	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
42	333	508824	407.059	317	484376	387.501	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
43	333	508824	407.059	269	411032	328.826	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
44	333	508824	407.059	311	475208	380.166	7	10696	8.5568	1	1528	1.2224	7	10696	8.5568	7	10696	8.5568
45	333	508824	407.059	290	443120	354.496	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
46	333	508824	407.059	262	400336	320.269	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
47	333	508824	407.059	275	420200	336.16	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
48	333	508824	407.059	328	501184	400.947	7	10696	8.5568	2	3056	2.4448	7	10696	8.5568	6	9168	7.3344
49	333	508824	407.059	319	487432	389.946	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
50	333	508824	407.059	314	479792	383.834	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
51	333	508824	407.059	280	427840	342.272	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
52	333	508824	407.059	309	472152	377.722	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	6	9168	7.3344
53	333	508824	407.059	321	490488	392.39	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
54	333	508824	407.059	259	395752	316.602	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
55	333	508824	407.059	269	411032	328.826	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
56	333	508824	407.059	292	446176	356.941	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
57	333	508824	407.059	291	444648	355.718	7	10696	8.5568	4	6112	4.8896	7	10696	8.5568	7	10696	8.5568
58	333	508824	407.059	281	429368	343.494	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
59	333	508824	407.059	321	490488	392.39	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
60	333	508824	407.059	270	412560	330.048	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
61	333	508824	407.059	319	487432	389.946	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
62	333	508824	407.059	273	417144	333.715	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
63	333	508824	407.059	275	420200	336.16	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
64	333	508824	407.059	286	437008	349.606	7	10696	8.5568	3	4584	3.6672	7	10696	8.5568	6	9168	7.3344
65	333	508824	407.059	245	374360	299.488	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
66	333	508824	407.059	303	462984	370.387	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	2	3056	2.4448
67	333	508824	407.059	325	496600	397.28	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
68	333	508824	407.059	284	433952	347.162	7	10696	8.5568	2	3056	2.4448	7	10696	8.5568	7	10696	8.5568
69	333	508824	407.059	318	485904	388.723	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
70	333	508824	407.059	248	378944	303.155	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
71	333	508824	407.059	320	488960	391.168	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
72	333	508824	407.059	260	397280	317.824	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
73	333	508824	407.059	268	409504	327.603	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
74	333	508824	407.059	307	469096	375.277	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
75	333	508824	407.059	327	499656	399.725	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	6	9168	7.3344
76	333	508824	407.059	261	398808	319.046	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
77	333	508824	407.059	305	466040	372.832	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	6	9168	7.3344
78	333	508824	407.059	250	382000	305.6	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
79	333	508824	407.059	321	490488	392.39	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
80	333	508824	407.059	329	502712	402.17	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
81	333	508824	407.059	329	502712	402.17	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
82	333	508824	407.059	328	501184	400.947	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	6	9168	7.3344
83	333	508824	407.059	331	505768	404.614	7	10696	8.5568	4	6112	4.8896	7	10696	8.5568	7	10696	8.5568
84	333	508824	407.059	296	452288	361.83	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
85	333	508824	407.059	319	487432	389.946	7	10696	8.5568	4	6112	4.8896	7	10696	8.5568	4	6112	4.8896
86	333	508824	407.059	314	479792	383.834	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	0	0	0
87	333	508824	407.059	300	458400	366.72	7	10696	8.5568	1	1528	1.2224	7	10696	8.5568	7	10696	8.5568
88	333	508824	407.059	315	481320	385.056	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
89	333	508824	407.059	283	432424	345.939	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
90	333	508824	407.059	295	450760	360.608	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
91	333	508824	407.059	272	415616	332.493	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
92	333	508824	407.059	278	424784	339.827	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
93	333	508824	407.059	291	444648	355.718	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	6	9168	7.3344
94	333	508824	407.059	326	498128	398.502	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344
95	333	508824	407.059	293	447704	358.163	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
96	333	508824	407.059	323	493544	394.835	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	7	10696	8.5568
97	333	508824	407.059	295	450760	360.608	7	10696	8.5568	5	7640	6.112	7	10696	8.5568	7	10696	8.5568
98	333	508824	407.059	300	458400	366.72	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	5	7640	6.112
99	333	508824	407.059	245	374360	299.488	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568	7	10696	8.5568
100	333	508824	407.059	315	481320	385.056	7	10696	8.5568	6	9168	7.3344	7	10696	8.5568	6	9168	7.3344

Table B.10: Interference 0.1 second

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	3333	5092824	407.426	3309	5056152	404.492	82	125296	10.0237	81	123768	9.90144	82	125296	10.0237	81	123768	9.90144
2	3333	5092824	407.426	3260	4981280	398.502	82	125296	10.0237	81	123768	9.90144	82	125296	10.0237	82	125296	10.0237
3	3333	5092824	407.426	3167	4839176	387.134	82	125296	10.0237	82	125296	10.0237	82	125296	10.0237	80	122240	9.7792
4	3333	5092824	407.426	3073	4695544	375.644	82	125296	10.0237	82	125296	10.0237	82	125296	10.0237	79	120712	9.65696
5	3333	5092824	407.426	3195	4881960	390.557	82	125296	10.0237	82	125296	10.0237	82	125296	10.0237	81	123768	9.90144
6	3333	5092824	407.426	3311	5059208	404.737	82	125296	10.0237	81	123768	9.90144	82	125296	10.0237	82	125296	10.0237
7	3333	5092824	407.426	3259	4979752	398.38	82	125296	10.0237	82	125296	10.0237	82	125296	10.0237	81	123768	9.90144
8	3333	5092824	407.426	3281	5013368	401.069	82	125296	10.0237	81	123768	9.90144	82	125296	10.0237	82	125296	10.0237
9	3333	5092824	407.426	3261	4982808	398.625	82	125296	10.0237	80	122240	9.7792	82	125296	10.0237	82	125296	10.0237
10	3333	5092824	407.426	3312	5060736	404.859	82	125296	10.0237	81	123768	9.90144	82	125296	10.0237	82	125296	10.0237
11	3333	5092824	407.426	3283	5016424	401.314	82	125296	10.0237	82	125296	10.0237	82	125296	10.0237	78	119184	9.53472
12	3333	5092824	407.426	3319	5071432	405.715	82	125296	10.0237	80	122240	9.7792	82	125296	10.0237	76	116128	9.29024

Table B.11: Interference 0.2 second

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
6	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
7	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
8	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
11	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.12: Interference 0.5 second

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	16666	25465648	407.45	16606	25373968	405.983	415	634120	10.1459	412	629536	10.0726	415	634120	10.1459	415	634120	10.1459
2	16666	25465648	407.45	16655	25448840	407.181	415	634120	10.1459	414	632592	10.1215	415	634120	10.1459	414	632592	10.1215
3	16666	25465648	407.45	16613	25384664	406.155	415	634120	10.1459	415	634120	10.1459	415	634120	10.1459	415	634120	10.1459
4	16666	25465648	407.45	16605	25372440	405.959	415	634120	10.1459	415	634120	10.1459	415	634120	10.1459	411	628008	10.0481
5	16666	25465648	407.45	16664	25462592	407.401	415	634120	10.1459	414	632592	10.1215	415	634120	10.1459	415	634120	10.1459
6	16666	25465648	407.45	16613	25384664	406.155	415	634120	10.1459	415	634120	10.1459	415	634120	10.1459	415	634120	10.1459

Table B.13: Interference 1 second

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	33333	50932824	407.463	33317	50908376	407.267	832	1271296	10.1704	832	1271296	10.1704	832	1271296	10.1704	831	1269768	10.1581
2	33333	50932824	407.463	33327	50923656	407.389	832	1271296	10.1704	827	1263656	10.1092	832	1271296	10.1704	831	1269768	10.1581
3	33333	50932824	407.463	33143	50642504	405.14	832	1271296	10.1704	830	1268240	10.1459	832	1271296	10.1704	830	1268240	10.1459

Table B.14: Interfering 100

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6656	10170368	406.815												
2	6666	10185648	407.426	6661	10178008	407.12												
3	6666	10185648	407.426	6652	10164256	406.57												
4	6666	10185648	407.426	6656	10170368	406.815												
5	6666	10185648	407.426	6650	10161200	406.448												
6	6666	10185648	407.426	6665	10184120	407.365												
7	6666	10185648	407.426	6649	10159672	406.387												
8	6666	10185648	407.426	6661	10178008	407.12												
9	6666	10185648	407.426	6665	10184120	407.365												
10	6666	10185648	407.426	6661	10178008	407.12												
11	6666	10185648	407.426	6658	10173424	406.937												
12	6666	10185648	407.426	6659	10174952	406.998												
13	6666	10185648	407.426	6618	10112304	404.492												
14	6666	10185648	407.426	6665	10184120	407.365												
15	6666	10185648	407.426	6662	10179536	407.181												
16	6666	10185648	407.426	6656	10170368	406.815												
17	6666	10185648	407.426	6655	10168840	406.754												
18	6666	10185648	407.426	6662	10179536	407.181												
19	6666	10185648	407.426	6654	10167312	406.692												
20	6666	10185648	407.426	6650	10161200	406.448												
21	6666	10185648	407.426	6648	10158144	406.326												
22	6666	10185648	407.426	6650	10161200	406.448												
23	6666	10185648	407.426	6616	10109248	404.37												
24	6666	10185648	407.426	6662	10179536	407.181												
25	6666	10185648	407.426	6664	10182592	407.304												
26	6666	10185648	407.426	6653	10165784	406.631												
27	6666	10185648	407.426	6662	10179536	407.181												
28	6666	10185648	407.426	6661	10178008	407.12												
29	6666	10185648	407.426	6649	10159672	406.387												
30	6666	10185648	407.426	6662	10179536	407.181												
31	6666	10185648	407.426	6660	10176480	407.059												
32	6666	10185648	407.426	6660	10176480	407.059												
33	6666	10185648	407.426	6662	10179536	407.181												
34	6666	10185648	407.426	6655	10168840	406.754												
35	6666	10185648	407.426	6660	10176480	407.059												
36	6666	10185648	407.426	6662	10179536	407.181												
37	6666	10185648	407.426	6662	10179536	407.181												
38	6666	10185648	407.426	6660	10176480	407.059												
39	6666	10185648	407.426	6660	10176480	407.059												
40	6666	10185648	407.426	6652	10164256	406.57												
41	6666	10185648	407.426	6663	10181064	407.243												
42	6666	10185648	407.426	6663	10181064	407.243												
43	6666	10185648	407.426	6664	10182592	407.304												
44	6666	10185648	407.426	6661	10178008	407.12												
45	6666	10185648	407.426	6666	10185648	407.426												
46	6666	10185648	407.426	6660	10176480	407.059												
47	6666	10185648	407.426	6647	10156616	406.265												
48	6666	10185648	407.426	6654	10167312	406.692												
49	6666	10185648	407.426	6659	10174952	406.998												
50	6666	10185648	407.426	6662	10179536	407.181												
51	6666	10185648	407.426	6651	10162728	406.509												
52	6666	10185648	407.426	6661	10178008	407.12												
53	6666	10185648	407.426	6629	10129112	405.164												
54	6666	10185648	407.426	6659	10174952	406.998												
55	6666	10185648	407.426	6661	10178008	407.12												
56	6666	10185648	407.426	6660	10176480	407.059												
57	6666	10185648	407.426	6652	10164256	406.57												
58	6666	10185648	407.426	6657	10171896	406.876												
59	6666	10185648	407.426	6650	10161200	406.448												
60	6666	10185648	407.426	6660	10176480	407.059												
61	6666	10185648	407.426	6660	10176480	407.059												
62	6666	10185648	407.426	6662	10179536	407.181												
63	6666	10185648	407.426	6651	10162728	406.509												
64	6666	10185648	407.426	6649	10159672	406.387												
65	6666	10185648	407.426	6660	10176480	407.059												
66	6666	10185648	407.426	6661	10178008	407.12												
67	6666	10185648	407.426	6663	10181064	407.243												
68	6666	10185648	407.426	6657	10171896	406.876												
69	6666	10185648	407.426	6663	10181064	407.243												
70	6666	10185648	407.426	6656	10170368	406.815												
71	6666	10185648	407.426	6665	10184120	407.365												
72	6666	10185648	407.426	6651	10162728	406.509												
73	6666	10185648	407.426	6656	10170368	406.815												
74	6666	10185648	407.426	6659	10174952	406.998												
75	6666	10185648	407.426	6650	10161200	406.448												
76	6666	10185648	407.426	6656	10170368	406.815												
77	6666	10185648	407.426	6657	10171896	406.876												
78	6666	10185648	407.426	6664	10182592	407.304												
79	6666	10185648	407.426	6661	10178008	407.12												
80	6666	10185648	407.426	6663	10181064	407.243												
81	6666	10185648	407.426	6659	10174952	406.998												
82	6666	10185648	407.426	6653	10165784	406.631												
83	6666	10185648	407.426	6662	10179536	407.181												
84	6666	10185648	407.426	6658	10173424	406.937												
85	6666	10185648	407.426	6666	10185648	407.426												
86	6666	10185648	407.426	6653	10165784	406.631												
87	6666	10185648	407.426	6663	10181064	407.243												
88	6666	10185648	407.426	6658	10173424	406.937												
89	6666	10185648	407.426	6666	10185648	407.426												
90	6666	10185648	407.426	6658	10173424	406.937												
91	6666	10185648	407.426	6661	10178008	407.12												
92	6666	10185648	407.426	6666	10185648	407.426												
93	6666	10185648	407.426	6661	10178008	407.12												
94	6666	10185648	407.426	6658	10173424	406.937												
95	6666	10185648	407.426	6650	10161200	406.448												
96	6666	10185648	407.426	6661	10178008	407.12												
97	6666	10185648	407.426	6651	10162728	406.509												
98	6666	10185648	407.426	6663	10181064	407.243												
99	6666	10185648	407.426	6660	10176480	407.059												
100	6666	10185648	407.426	6663	10181064	407.243												

Table B.15: Interfering 1 000

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6656	10170368	406.815												
2	6666	10185648	407.426	6661	10178008	407.12												
3	6666	10185648	407.426	6652	10164256	406.57												
4	6666	10185648	407.426	6656	10170368	406.815												
5	6666	10185648	407.426	6650	10161200	406.448												
6	6666	10185648	407.426	6665	10184120	407.365												
7	6666	10185648	407.426	6649	10159672	406.387												
8	6666	10185648	407.426	6661	10178008	407.12												
9	6666	10185648	407.426	6665	10184120	407.365												
10	6666	10185648	407.426	6661	10178008	407.12												
11	6666	10185648	407.426	6658	10173424	406.937												
12	6666	10185648	407.426	6659	10174952	406.998												
13	6666	10185648	407.426	6618	10112304	404.492												
14	6666	10185648	407.426	6665	10184120	407.365												
15	6666	10185648	407.426	6662	10179536	407.181												
16	6666	10185648	407.426	6656	10170368	406.815												
17	6666	10185648	407.426	6655	10168840	406.754												
18	6666	10185648	407.426	6662	10179536	407.181												
19	6666	10185648	407.426	6654	10167312	406.692												
20	6666	10185648	407.426	6650	10161200	406.448												
21	6666	10185648	407.426	6648	10158144	406.326												
22	6666	10185648	407.426	6650	10161200	406.448												
23	6666	10185648	407.426	6616	10109248	404.37												
24	6666	10185648	407.426	6662	10179536	407.181												
25	6666	10185648	407.426	6664	10182592	407.304												
26	6666	10185648	407.426	6653	10165784	406.631												
27	6666	10185648	407.426	6662	10179536	407.181												
28	6666	10185648	407.426	6661	10178008	407.12												
29	6666	10185648	407.426	6649	10159672	406.387												
30	6666	10185648	407.426	6662	10179536	407.181												
31	6666	10185648	407.426	6660	10176480	407.059												
32	6666	10185648	407.426	6660	10176480	407.059												
33	6666	10185648	407.426	6662	10179536	407.181												
34	6666	10185648	407.426	6655	10168840	406.754												
35	6666	10185648	407.426	6660	10176480	407.059												
36	6666	10185648	407.426	6662	10179536	407.181												
37	6666	10185648	407.426	6662	10179536	407.181												
38	6666	10185648	407.426	6660	10176480	407.059												
39	6666	10185648	407.426	6660	10176480	407.059												
40	6666	10185648	407.426	6652	10164256	406.57												
41	6666	10185648	407.426	6663	10181064	407.243												
42	6666	10185648	407.426	6663	10181064	407.243												
43	6666	10185648	407.426	6664	10182592	407.304												
44	6666	10185648	407.426	6661	10178008	407.12												
45	6666	10185648	407.426	6666	10185648	407.426												
46	6666	10185648	407.426	6660	10176480	407.059												
47	6666	10185648	407.426	6647	10156616	406.265												
48	6666	10185648	407.426	6654	10167312	406.692												
49	6666	10185648	407.426	6659	10174952	406.998												
50	6666	10185648	407.426	6662	10179536	407.181												
51	6666	10185648	407.426	6651	10162728	406.509												
52	6666	10185648	407.426	6661	10178008	407.12												
53	6666	10185648	407.426	6629	10129112	405.164												
54	6666	10185648	407.426	6659	10174952	406.998												
55	6666	10185648	407.426	6661	10178008	407.12												
56	6666	10185648	407.426	6660	10176480	407.059												
57	6666	10185648	407.426	6652	10164256	406.57												
58	6666	10185648	407.426	6657	10171896	406.876												
59	6666	10185648	407.426	6650	10161200	406.448												
60	6666	10185648	407.426	6660	10176480	407.059												
61	6666	10185648	407.426	6660	10176480	407.059												
62	6666	10185648	407.426	6662	10179536	407.181												
63	6666	10185648	407.426	6651	10162728	406.509												
64	6666	10185648	407.426	6649	10159672	406.387												
65	6666	10185648	407.426	6660	10176480	407.059												
66	6666	10185648	407.426	6661	10178008	407.12												
67	6666	10185648	407.426	6663	10181064	407.243												
68	6666	10185648	407.426	6657	10171896	406.876												
69	6666	10185648	407.426	6663	10181064	407.243												
70	6666	10185648	407.426	6656	10170368	406.815												
71	6666	10185648	407.426	6665	10184120	407.365												
72	6666	10185648	407.426	6651	10162728	406.509												
73	6666	10185648	407.426	6656	10170368	406.815												
74	6666	10185648	407.426	6659	10174952	406.998												
75	6666	10185648	407.426	6650	10161200	406.448												
76	6666	10185648	407.426	6656	10170368	406.815												
77	6666	10185648	407.426	6657	10171896	406.876												
78	6666	10185648	407.426	6664	10182592	407.304												
79	6666	10185648	407.426	6661	10178008	407.12												
80	6666	10185648	407.426	6663	10181064	407.243												
81	6666	10185648	407.426	6659	10174952	406.998												
82	6666	10185648	407.426	6653	10165784	406.631												
83	6666	10185648	407.426	6662	10179536	407.181												
84	6666	10185648	407.426	6658	10173424	406.937												
85	6666	10185648	407.426	6666	10185648	407.426												
86	6666	10185648	407.426	6653	10165784	406.631												
87	6666	10185648	407.426	6663	10181064	407.243												
88	6666	10185648	407.426	6658	10173424	406.937												
89	6666	10185648	407.426	6666	10185648	407.426												
90	6666	10185648	407.426	6658	10173424	406.937												
91	6666	10185648	407.426	6661	10178008	407.12												
92	6666	10185648	407.426	6666	10185648	407.426												
93	6666	10185648	407.426	6661	10178008	407.12												
94	6666	10185648	407.426	6658	10173424	406.937												
95	6666	10185648	407.426	6650	10161200	406.448												
96	6666	10185648	407.426	6661	10178008	407.12												
97	6666	10185648	407.426	6651	10162728	406.509												
98	6666	10185648	407.426	6663	10181064	407.243												
99	6666	10185648	407.426	6660	10176480	407.059												
100	6666	10185648	407.426	6663	10181064	407.243												

Table B.16: Interfering 10 000

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6656	10170368	406.815												
2	6666	10185648	407.426	6661	10178008	407.12												
3	6666	10185648	407.426	6652	10164256	406.57												
4	6666	10185648	407.426	6656	10170368	406.815												
5	6666	10185648	407.426	6650	10161200	406.448												
6	6666	10185648	407.426	6665	10184120	407.365												
7	6666	10185648	407.426	6649	10159672	406.387												
8	6666	10185648	407.426	6661	10178008	407.12												
9	6666	10185648	407.426	6665	10184120	407.365												
10	6666	10185648	407.426	6661	10178008	407.12												
11	6666	10185648	407.426	6658	10173424	406.937												
12	6666	10185648	407.426	6659	10174952	406.998												
13	6666	10185648	407.426	6618	10112304	404.492												
14	6666	10185648	407.426	6665	10184120	407.365												
15	6666	10185648	407.426	6662	10179536	407.181												
16	6666	10185648	407.426	6656	10170368	406.815												
17	6666	10185648	407.426	6655	10168840	406.754												
18	6666	10185648	407.426	6662	10179536	407.181												
19	6666	10185648	407.426	6654	10167312	406.692												
20	6666	10185648	407.426	6650	10161200	406.448												
21	6666	10185648	407.426	6648	10158144	406.326												
22	6666	10185648	407.426	6650	10161200	406.448												
23	6666	10185648	407.426	6616	10109248	404.37												
24	6666	10185648	407.426	6662	10179536	407.181												
25	6666	10185648	407.426	6664	10182592	407.304												
26	6666	10185648	407.426	6653	10165784	406.631												
27	6666	10185648	407.426	6662	10179536	407.181												
28	6666	10185648	407.426	6661	10178008	407.12												
29	6666	10185648	407.426	6649	10159672	406.387												
30	6666	10185648	407.426	6662	10179536	407.181												
31	6666	10185648	407.426	6660	10176480	407.059												
32	6666	10185648	407.426	6660	10176480	407.059												
33	6666	10185648	407.426	6662	10179536	407.181												
34	6666	10185648	407.426	6655	10168840	406.754												
35	6666	10185648	407.426	6660	10176480	407.059												
36	6666	10185648	407.426	6662	10179536	407.181												
37	6666	10185648	407.426	6662	10179536	407.181												
38	6666	10185648	407.426	6660	10176480	407.059												
39	6666	10185648	407.426	6660	10176480	407.059												
40	6666	10185648	407.426	6652	10164256	406.57												
41	6666	10185648	407.426	6663	10181064	407.243												
42	6666	10185648	407.426	6663	10181064	407.243												
43	6666	10185648	407.426	6664	10182592	407.304												
44	6666	10185648	407.426	6661	10178008	407.12												
45	6666	10185648	407.426	6666	10185648	407.426												
46	6666	10185648	407.426	6660	10176480	407.059												
47	6666	10185648	407.426	6647	10156616	406.265												
48	6666	10185648	407.426	6654	10167312	406.692												
49	6666	10185648	407.426	6659	10174952	406.998												
50	6666	10185648	407.426	6662	10179536	407.181												
51	6666	10185648	407.426	6651	10162728	406.509												
52	6666	10185648	407.426	6661	10178008	407.12												
53	6666	10185648	407.426	6629	10129112	405.164												
54	6666	10185648	407.426	6659	10174952	406.998												
55	6666	10185648	407.426	6661	10178008	407.12												
56	6666	10185648	407.426	6660	10176480	407.059												
57	6666	10185648	407.426	6652	10164256	406.57												
58	6666	10185648	407.426	6657	10171896	406.876												
59	6666	10185648	407.426	6650	10161200	406.448												
60	6666	10185648	407.426	6660	10176480	407.059												
61	6666	10185648	407.426	6660	10176480	407.059												
62	6666	10185648	407.426	6662	10179536	407.181												
63	6666	10185648	407.426	6651	10162728	406.509												
64	6666	10185648	407.426	6649	10159672	406.387												
65	6666	10185648	407.426	6660	10176480	407.059												
66	6666	10185648	407.426	6661	10178008	407.12												
67	6666	10185648	407.426	6663	10181064	407.243												
68	6666	10185648	407.426	6657	10171896	406.876												
69	6666	10185648	407.426	6663	10181064	407.243												
70	6666	10185648	407.426	6656	10170368	406.815												
71	6666	10185648	407.426	6665	10184120	407.365												
72	6666	10185648	407.426	6651	10162728	406.509												
73	6666	10185648	407.426	6656	10170368	406.815												
74	6666	10185648	407.426	6659	10174952	406.998												
75	6666	10185648	407.426	6650	10161200	406.448												
76	6666	10185648	407.426	6656	10170368	406.815												
77	6666	10185648	407.426	6657	10171896	406.876												
78	6666	10185648	407.426	6664	10182592	407.304												
79	6666	10185648	407.426	6661	10178008	407.12												
80	6666	10185648	407.426	6663	10181064	407.243												
81	6666	10185648	407.426	6659	10174952	406.998												
82	6666	10185648	407.426	6653	10165784	406.631												
83	6666	10185648	407.426	6662	10179536	407.181												
84	6666	10185648	407.426	6658	10173424	406.937												
85	6666	10185648	407.426	6666	10185648	407.426												
86	6666	10185648	407.426	6653	10165784	406.631												
87	6666	10185648	407.426	6663	10181064	407.243												
88	6666	10185648	407.426	6658	10173424	406.937												
89	6666	10185648	407.426	6666	10185648	407.426												
90	6666	10185648	407.426	6658	10173424	406.937												
91	6666	10185648	407.426	6661	10178008	407.12												
92	6666	10185648	407.426	6666	10185648	407.426												
93	6666	10185648	407.426	6661	10178008	407.12												
94	6666	10185648	407.426	6658	10173424	406.937												
95	6666	10185648	407.426	6650	10161200	406.448												
96	6666	10185648	407.426	6661	10178008	407.12												
97	6666	10185648	407.426	6651	10162728	406.509												
98	6666	10185648	407.426	6663	10181064	407.243												
99	6666	10185648	407.426	6660	10176480	407.059												
100	6666	10185648	407.426	6663	10181064	407.243												

Table B.17: Interfering 100 000

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6646	10155088	406.204	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
2	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
3	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
4	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
5	6666	10185648	407.426	6655	10168840	406.754	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
6	6666	10185648	407.426	6657	10171896	406.876	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
7	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
8	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
9	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
10	6666	10185648	407.426	6640	10145920	405.837	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
11	6666	10185648	407.426	6652	10164256	406.57	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
12	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
13	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
14	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
15	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
16	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
17	6666	10185648	407.426	6652	10164256	406.57	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
18	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
19	6666	10185648	407.426	6659	10174952	406.998	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
20	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
21	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
22	6666	10185648	407.426	6652	10164256	406.57	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
23	6666	10185648	407.426	6644	10152032	406.081	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
24	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
25	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
26	6666	10185648	407.426	6659	10174952	406.998	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
27	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
28	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
29	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
30	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
31	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
32	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
33	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
34	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
35	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
36	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
37	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
38	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
39	6666	10185648	407.426	6652	10164256	406.57	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
40	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
41	6666	10185648	407.426	6654	10167312	406.692	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
42	6666	10185648	407.426	6658	10173424	406.937	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
43	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
44	6666	10185648	407.426	6654	10167312	406.692	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
45	6666	10185648	407.426	6656	10170368	406.815	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
46	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
47	6666	10185648	407.426	6654	10167312	406.692	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
48	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
49	6666	10185648	407.426	6650	10161200	406.448	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
50	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
51	6666	10185648	407.426	6632	10133696	405.348	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
52	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
53	6666	10185648	407.426	6637	10141376	405.653	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
54	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
55	6666	10185648	407.426	6658	10173424	406.937	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
56	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
57	6666	10185648	407.426	6659	10174952	406.998	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
58	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
59	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
60	6666	10185648	407.426	6650	10161200	406.448	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
61	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
62	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
63	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
64	6666	10185648	407.426	6659	10174952	406.998	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
65	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
66	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
67	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
68	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
69	6666	10185648	407.426	6650	10161200	406.448	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
70	6666	10185648	407.426	6659	10174952	406.998	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
71	6666	10185648	407.426	6657	10171896	406.876	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
72	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
73	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
74	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
75	6666	10185648	407.426	6651	10162728	406.509	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
76	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
77	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
78	6666	10185648	407.426	6656	10170368	406.815	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
79	6666	10185648	407.426	6651	10162728	406.509	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
80	6666	10185648	407.426	6660	10176480	407.059	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
81	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
82	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
83	6666	10185648	407.426	6653	10165784	406.631	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
84	6666	10185648	407.426	6655	10168840	406.754	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
85	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
86	6666	10185648	407.426	6652	10164256	406.57	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
87	6666	10185648	407.426	6658	10173424	406.937	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
88	6666	10185648	407.426	6639	10144392	405.776	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
89	6666	10185648	407.426	6664	10182592	407.304	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
90	6666	10185648	407.426	6662	10179536	407.181	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
91	6666	10185648	407.426	6663	10181064	407.243	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
92	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
93	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
94	6666	10185648	407.426	6654	10167312	406.692	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
95	6666	10185648	407.426	6657	10171896	406.876	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
96	6666	10185648	407.426	6666	10185648	407.426	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
97	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
98	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
99	6666	10185648	407.426	6661	10178008	407.12	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112
100	6666	10185648	407.426	6665	10184120	407.365	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112	1	1528	0.06112

Table B.18: Interfering 1M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6657	10171896	406.876	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
2	6666	10185648	407.426	6664	10182592	407.304	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
3	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
4	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
5	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
6	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
7	6666	10185648	407.426	6660	10176480	407.059	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
8	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
9	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
10	6666	10185648	407.426	6651	10162728	406.509	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
11	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
12	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
13	6666	10185648	407.426	6664	10182592	407.304	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
14	6666	10185648	407.426	6650	10161200	406.448	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
15	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
16	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
17	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
18	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
19	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
20	6666	10185648	407.426	6659	10174952	406.998	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
21	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
22	6666	10185648	407.426	6664	10182592	407.304	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
23	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
24	6666	10185648	407.426	6659	10174952	406.998	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
25	6666	10185648	407.426	6656	10170368	406.815	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
26	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
27	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
28	6666	10185648	407.426	6649	10159672	406.387	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
29	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
30	6666	10185648	407.426	6652	10164256	406.57	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
31	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
32	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
33	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
34	6666	10185648	407.426	6652	10164256	406.57	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
35	6666	10185648	407.426	6664	10182592	407.304	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
36	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
37	6666	10185648	407.426	6647	10156616	406.265	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
38	6666	10185648	407.426	6658	10173424	406.937	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
39	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
40	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
41	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
42	6666	10185648	407.426	6657	10171896	406.876	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
43	6666	10185648	407.426	6656	10170368	406.815	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
44	6666	10185648	407.426	6657	10171896	406.876	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
45	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
46	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
47	6666	10185648	407.426	6652	10164256	406.57	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
48	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
49	6666	10185648	407.426	6651	10162728	406.509	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
50	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
51	6666	10185648	407.426	6637	10141336	405.653	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
52	6666	10185648	407.426	6662	10179536	407.181	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
53	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
54	6666	10185648	407.426	6653	10165784	406.631	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
55	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
56	6666	10185648	407.426	6650	10161200	406.448	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
57	6666	10185648	407.426	6635	10138280	405.531	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
58	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
59	6666	10185648	407.426	6649	10159672	406.387	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
60	6666	10185648	407.426	6635	10138280	405.531	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
61	6666	10185648	407.426	6653	10165784	406.631	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
62	6666	10185648	407.426	6660	10176480	407.059	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
63	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
64	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
65	6666	10185648	407.426	6660	10176480	407.059	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
66	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
67	6666	10185648	407.426	6650	10161200	406.448	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
68	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
69	6666	10185648	407.426	6659	10174952	406.998	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
70	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
71	6666	10185648	407.426	6659	10174952	406.998	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
72	6666	10185648	407.426	6658	10173424	406.937	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
73	6666	10185648	407.426	6653	10165784	406.631	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
74	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
75	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
76	6666	10185648	407.426	6666	10185648	407.426	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
77	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
78	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
79	6666	10185648	407.426	6656	10170368	406.815	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
80	6666	10185648	407.426	6656	10170368	406.815	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
81	6666	10185648	407.426	6658	10173424	406.937	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
82	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
83	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
84	6666	10185648	407.426	6658	10173424	406.937	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
85	6666	10185648	407.426	6655	10168840	406.754	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
86	6666	10185648	407.426	6654	10167312	406.692	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
87	6666	10185648	407.426	6663	10181064	407.243	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
88	6666	10185648	407.426	6652	10164256	406.57	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
89	6666	10185648	407.426	6657	10171896	406.876	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
90	6666	10185648	407.426	6661	10178008	407.12	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792
91	6666	10185648	407.426	6665	10184120	407.365	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792	16	24448	0.97792

Table B.19: Interfering 2M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6658	10173424	406.937	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
2	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
3	6666	10185648	407.426	6652	10164256	406.57	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
4	6666	10185648	407.426	6646	10155088	406.204	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
5	6666	10185648	407.426	6647	10156616	406.265	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
6	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
7	6666	10185648	407.426	6632	10133696	405.348	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
8	6666	10185648	407.426	6624	10121472	404.859	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
9	6666	10185648	407.426	6657	10171896	406.876	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
10	6666	10185648	407.426	6627	10126056	405.042	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
11	6666	10185648	407.426	6650	10161200	406.448	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
12	6666	10185648	407.426	6647	10156616	406.265	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
13	6666	10185648	407.426	6656	10170368	406.815	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
14	6666	10185648	407.426	6649	10159672	406.387	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
15	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
16	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
17	6666	10185648	407.426	6623	10119944	404.798	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
18	6666	10185648	407.426	6663	10181064	407.243	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
19	6666	10185648	407.426	6661	10178008	407.12	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
20	6666	10185648	407.426	6632	10133696	405.348	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
21	6666	10185648	407.426	6637	10141336	405.653	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
22	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
23	6666	10185648	407.426	6621	10116888	404.676	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
24	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
25	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
26	6666	10185648	407.426	6631	10132168	405.287	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
27	6666	10185648	407.426	6628	10127584	405.103	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
28	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
29	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
30	6666	10185648	407.426	6624	10121472	404.859	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
31	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
32	6666	10185648	407.426	6662	10179536	407.181	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
33	6666	10185648	407.426	6637	10141336	405.653	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
34	6666	10185648	407.426	6632	10133696	405.348	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
35	6666	10185648	407.426	6652	10164256	406.57	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
36	6666	10185648	407.426	6641	10147448	405.898	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
37	6666	10185648	407.426	6653	10165784	406.631	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
38	6666	10185648	407.426	6651	10162728	406.509	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
39	6666	10185648	407.426	6651	10162728	406.509	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
40	6666	10185648	407.426	6658	10173424	406.937	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
41	6666	10185648	407.426	6661	10178008	407.12	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
42	6666	10185648	407.426	6658	10173424	406.937	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
43	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
44	6666	10185648	407.426	6653	10165784	406.631	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
45	6666	10185648	407.426	6618	10112304	404.492	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
46	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
47	6666	10185648	407.426	6645	10153560	406.142	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	31	47368	1.89472
48	6666	10185648	407.426	6653	10165784	406.631	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
49	6666	10185648	407.426	6637	10141336	405.653	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
50	6666	10185648	407.426	6663	10181064	407.243	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
51	6666	10185648	407.426	6637	10141336	405.653	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
52	6666	10185648	407.426	6618	10112304	404.492	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
53	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
54	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
55	6666	10185648	407.426	6611	10101608	404.064	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
56	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
57	6666	10185648	407.426	6632	10133696	405.348	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
58	6666	10185648	407.426	6650	10161200	406.448	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
59	6666	10185648	407.426	6665	10184120	407.365	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
60	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
61	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
62	6666	10185648	407.426	6662	10179536	407.181	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
63	6666	10185648	407.426	6651	10162728	406.509	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
64	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
65	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
66	6666	10185648	407.426	6654	10167312	406.692	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
67	6666	10185648	407.426	6661	10178008	407.12	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
68	6666	10185648	407.426	6650	10161200	406.448	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
69	6666	10185648	407.426	6630	10130640	405.226	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
70	6666	10185648	407.426	6663	10181064	407.243	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	31	47368	1.89472
71	6666	10185648	407.426	6648	10158144	406.326	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
72	6666	10185648	407.426	6640	10145920	405.837	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
73	6666	10185648	407.426	6662	10179536	407.181	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
74	6666	10185648	407.426	6654	10167312	406.692	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
75	6666	10185648	407.426	6663	10181064	407.243	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
76	6666	10185648	407.426	6625	10123000	404.92	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
77	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
78	6666	10185648	407.426	6647	10156616	406.265	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
79	6666	10185648	407.426	6648	10158144	406.326	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	30	45840	1.8336
80	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
81	6666	10185648	407.426	6621	10116888	404.676	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
82	6666	10185648	407.426	6639	10144392	405.776	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
83	6666	10185648	407.426	6615	10107720	404.309	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
84	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
85	6666	10185648	407.426	6659	10174952	406.998	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
86	6666	10185648	407.426	6627	10126056	405.042	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
87	6666	10185648	407.426	6632	10133696	405.348	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
88	6666	10185648	407.426	6647	10156616	406.265	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
89	6666	10185648	407.426	6638	10142864	405.715	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
90	6666	10185648	407.426	6663	10181064	407.243	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
91	6666	10185648	407.426	6648	10158144	406.326	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
92	6666	10185648	407.426	6661	10178008	407.12	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
93	6666	10185648	407.426	6660	10176480	407.059	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	32	48896	1.95584
94	6666	10185648	407.426	6655	10168840	406.754	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584
95	6666	10185648	407.426	6626	10124528	404.981	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	33	50424	2.01696
96	6666	10185648	407.426	6637	10141336	405.653	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
97	6666	10185648	407.426	6635	10138280	405.531	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
98	6666	10185648	407.426	6653	10165784	406.631	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696	33	50424	2.01696
99	6666	10185648	407.426	6662	10179536	407.181	33	50424	2.01696	32	48896	1.95584	33	50424	2.01696	32	48896	1.95584

Table B.20: Interfering 5M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6640	10145920	405.837	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
2	6666	10185648	407.426	6649	10159672	406.387	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
3	6666	10185648	407.426	6624	10121472	404.859	82	125296	5.01184	81	123768	4.95072	82	125296	5.01184	82	125296	5.01184
4	6666	10185648	407.426	6658	10173424	406.937	82	125296	5.01184	81	123768	4.95072	82	125296	5.01184	82	125296	5.01184
5	6666	10185648	407.426	6652	10164256	406.57	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
6	6666	10185648	407.426	6642	10148976	405.959	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
7	6666	10185648	407.426	6662	10179536	407.181	82	125296	5.01184	81	123768	4.95072	82	125296	5.01184	82	125296	5.01184
8	6666	10185648	407.426	6629	10129112	405.164	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	81	123768	4.95072
9	6666	10185648	407.426	6666	10185648	407.426	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
10	6666	10185648	407.426	6638	10142864	405.715	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
11	6666	10185648	407.426	6546	10002288	400.092	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
12	6666	10185648	407.426	6647	10156616	406.265	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
13	6666	10185648	407.426	6666	10185648	407.426	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
14	6666	10185648	407.426	6665	10184120	407.365	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
15	6666	10185648	407.426	6664	10182592	407.304	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
16	6666	10185648	407.426	6641	10147448	405.898	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
17	6666	10185648	407.426	6657	10171896	406.876	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
18	6666	10185648	407.426	6661	10178008	407.12	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
19	6666	10185648	407.426	6649	10159672	406.387	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
20	6666	10185648	407.426	6664	10182592	407.304	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
21	6666	10185648	407.426	6656	10170368	406.815	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
22	6666	10185648	407.426	6637	10141336	405.653	82	125296	5.01184	80	122240	4.8896	82	125296	5.01184	82	125296	5.01184
23	6666	10185648	407.426	6630	10130640	405.226	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
24	6666	10185648	407.426	6624	10121472	404.859	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	78	119184	4.76736
25	6666	10185648	407.426	6658	10173424	406.937	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
26	6666	10185648	407.426	6662	10179536	407.181	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
27	6666	10185648	407.426	6658	10173424	406.937	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
28	6666	10185648	407.426	6653	10165784	406.631	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	81	123768	4.95072
29	6666	10185648	407.426	6658	10173424	406.937	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
30	6666	10185648	407.426	6650	10161200	406.448	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
31	6666	10185648	407.426	6627	10126056	405.042	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
32	6666	10185648	407.426	6651	10162728	406.509	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
33	6666	10185648	407.426	6645	10153560	406.142	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	81	123768	4.95072
34	6666	10185648	407.426	6659	10174952	406.998	82	125296	5.01184	81	123768	4.95072	82	125296	5.01184	82	125296	5.01184
35	6666	10185648	407.426	6647	10156616	406.265	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
36	6666	10185648	407.426	6652	10164256	406.57	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
37	6666	10185648	407.426	6634	10136752	405.47	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
38	6666	10185648	407.426	6664	10182592	407.304	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
39	6666	10185648	407.426	6662	10179536	407.181	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
40	6666	10185648	407.426	6643	10150504	406.02	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
41	6666	10185648	407.426	6655	10168840	406.754	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
42	6666	10185648	407.426	6646	10155088	406.204	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
43	6666	10185648	407.426	6643	10150504	406.02	82	125296	5.01184	80	122240	4.8896	82	125296	5.01184	82	125296	5.01184
44	6666	10185648	407.426	6665	10184120	407.365	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
45	6666	10185648	407.426	6625	10123000	404.92	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	81	123768	4.95072
46	6666	10185648	407.426	6613	10104664	404.187	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
47	6666	10185648	407.426	6642	10148976	405.959	82	125296	5.01184	81	123768	4.95072	82	125296	5.01184	82	125296	5.01184
48	6666	10185648	407.426	6657	10171896	406.876	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
49	6666	10185648	407.426	6617	10110776	404.431	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184
50	6666	10185648	407.426	6666	10185648	407.426	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184	82	125296	5.01184

Table B.21: Interfering 10M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6633	10135224	405.409	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	162	247536	9.90144
4	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
6	6666	10185648	407.426	6638	10142864	405.715	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
7	6666	10185648	407.426	6550	10008400	400.336	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6640	10145920	405.837	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6658	10173424	406.937	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6660	10176480	407.059	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	165	252120	10.0848
11	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
12	6666	10185648	407.426	6651	10162728	406.509	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
13	6666	10185648	407.426	6576	10048128	401.925	165	252120	10.0848	159	242952	9.71808	165	252120	10.0848	164	250592	10.0237
14	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
15	6666	10185648	407.426	6643	10150504	406.02	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
16	6666	10185648	407.426	6651	10162728	406.509	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
17	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
18	6666	10185648	407.426	6630	10130640	405.226	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
19	6666	10185648	407.426	6606	10093968	403.759	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
20	6666	10185648	407.426	6652	10164256	406.57	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
21	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
22	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
23	6666	10185648	407.426	6575	10046600	401.864	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
24	6666	10185648	407.426	6586	10063408	402.536	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
25	6666	10185648	407.426	6578	10051184	402.047	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
26	6666	10185648	407.426	6515	9954920	398.197	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
27	6666	10185648	407.426	6598	10081744	403.27	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	163	249064	9.96256
28	6666	10185648	407.426	6505	9939640	397.586	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
29	6666	10185648	407.426	6645	10153560	406.142	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	160	244480	9.7792
30	6666	10185648	407.426	6601	10086328	403.453	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
31	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
32	6666	10185648	407.426	6529	9976312	399.052	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	161	246008	9.84032
33	6666	10185648	407.426	6652	10164256	406.57	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
34	6666	10185648	407.426	6645	10153560	406.142	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
35	6666	10185648	407.426	6636	10139808	405.592	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
36	6666	10185648	407.426	6654	10167312	406.692	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
37	6666	10185648	407.426	6665	10184120	407.365	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	163	249064	9.96256
38	6666	10185648	407.426	6589	10067992	402.72	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
39	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
40	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
41	6666	10185648	407.426	6573	10043544	401.742	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
42	6666	10185648	407.426	6599	10083272	403.331	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
43	6666	10185648	407.426	6648	10158144	406.326	165	252120	10.0848	162	247536	9.90144	165	252120	10.0848	165	252120	10.0848
44	6666	10185648	407.426	6571	10040488	401.62	165	252120	10.0848	162	247536	9.90144	165	252120	10.0848	165	252120	10.0848
45	6666	10185648	407.426	6586	10063408	402.536	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
46	6666	10185648	407.426	6636	10139808	405.592	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
47	6666	10185648	407.426	6312	9644736	385.789	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
48	6666	10185648	407.426	6556	10017568	400.703	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	163	249064	9.96256
49	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
50	6666	10185648	407.426	6614	10106192	404.248	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	163	249064	9.96256
51	6666	10185648	407.426	6590	10069520	402.781	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
52	6666	10185648	407.426	6522	9965616	398.625	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
53	6666	10185648	407.426	6587	10064936	402.597	165	252120	10.0848	160	244480	9.7792	165	252120	10.0848	165	252120	10.0848
54	6666	10185648	407.426	6631	10132168	405.287	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
55	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
56	6666	10185648	407.426	6609	10098552	403.942	165	252120	10.0848	162	247536	9.90144	165	252120	10.0848	165	252120	10.0848
57	6666	10185648	407.426	6529	9976312	399.052	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
58	6666	10185648	407.426	6652	10164256	406.57	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	165	252120	10.0848
59	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
60	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
61	6666	10185648	407.426	6644	10152032	406.081	165	252120	10.0848	162	247536	9.90144	165	252120	10.0848	165	252120	10.0848
62	6666	10185648	407.426	6566	10032848	401.314	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
63	6666	10185648	407.426	6637	10141336	405.653	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	161	246008	9.84032
64	6666	10185648	407.426	6659	10174952	406.998	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
65	6666	10185648	407.426	6664	10182592	407.304	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	159	242952	9.71808
66	6666	10185648	407.426	6623	10119944	404.798	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
67	6666	10185648	407.426	6632	10133696	405.348	165	252120	10.0848	160	244480	9.7792	165	252120	10.0848	165	252120	10.0848
68	6666	10185648	407.426	6658	10173424	406.937	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	164	250592	10.0237
69	6666	10185648	407.426	6637	10141336	405.653	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
70	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
71	6666	10185648	407.426	6635	10138280	405.531	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
72	6666	10185648	407.426	6663	10181064	407.243	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
73	6666	10185648	407.426	6622	10118416	404.737	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
74	6666	10185648	407.426	6658	10173424	406.937	165	252120	10.0848	159	242952	9.71808	165	252120	10.0848	165	252120	10.0848
75	6666	10185648	407.426	6626	10124528	404.981	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
76	6666	10185648	407.426	6558	10020624	400.825	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
77	6666	10185648	407.426	6610	10100080	404.003	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
78	6666	10185648	407.426	6654	10167312	406.692	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	163	249064	9.96256
79	6666	10185648	407.426	6660	10176480	407.059	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	163	249064	9.96256
80	6666	10185648	407.426	6631	10132168	405.287	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	163	249064	9.96256
81	6666	10185648	407.426	6531	9979368	399.175	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
82	6666	10185648	407.426	6338	9684464	387.379	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
83	6666	10185648	407.426	6410	9794480	391.779	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237
84	6666	10185648	407.426	6649	10159672	406.387	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
85	6666	10185648	407.426	6663	10181064	407.243	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
86	6666	10185648	407.426	6656	10170368	406.815	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
87	6666	10185648	407.426	6637	10141336	405.653	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
88	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	164	250592	10.0237

Table B.22: Interfering 20M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6592	10072576	402.903	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	326	498128	19.9251
2	6666	10185648	407.426	6209	9487352	379.494	331	505768	20.2307	328	501184	20.0474	331	505768	20.2307	330	504240	20.1696
3	6666	10185648	407.426	6597	10080216	403.209	331	505768	20.2307	326	498128	19.9251	331	505768	20.2307	328	501184	20.0474
4	6666	10185648	407.426	6343	9692104	387.684	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	329	502712	20.1085
5	6666	10185648	407.426	6654	10167312	406.692	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	330	504240	20.1696
6	6666	10185648	407.426	6384	9754752	390.19	331	505768	20.2307	326	498128	19.9251	331	505768	20.2307	331	505768	20.2307
7	6666	10185648	407.426	6176	9436928	377.477	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	323	493544	19.7418
8	6666	10185648	407.426	6652	10164256	406.57	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	331	505768	20.2307
9	6666	10185648	407.426	6561	10025208	401.008	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	330	504240	20.1696
10	6666	10185648	407.426	6287	9606536	384.261	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	330	504240	20.1696
11	6666	10185648	407.426	6561	10025208	401.008	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	329	502712	20.1085
12	6666	10185648	407.426	6409	9792952	391.718	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	330	504240	20.1696
13	6666	10185648	407.426	6638	10142864	405.715	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	329	502712	20.1085
14	6666	10185648	407.426	6154	9403312	376.132	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	330	504240	20.1696
15	6666	10185648	407.426	6653	10165784	406.631	331	505768	20.2307	325	496600	19.864	331	505768	20.2307	331	505768	20.2307
16	6666	10185648	407.426	6474	9892272	395.691	331	505768	20.2307	329	502712	20.1085	331	505768	20.2307	331	505768	20.2307
17	6666	10185648	407.426	5850	8938800	357.552	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307
18	6666	10185648	407.426	6577	10049656	401.986	331	505768	20.2307	324	495072	19.8029	331	505768	20.2307	330	504240	20.1696
19	6666	10185648	407.426	6159	9410952	376.438	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	330	504240	20.1696
20	6666	10185648	407.426	6497	9927416	397.097	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	329	502712	20.1085
21	6666	10185648	407.426	6664	10182592	407.304	331	505768	20.2307	324	495072	19.8029	331	505768	20.2307	321	490488	19.6195
22	6666	10185648	407.426	6567	10034376	401.375	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	327	499656	19.9862
23	6666	10185648	407.426	6337	9682936	387.317	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	329	502712	20.1085
24	6666	10185648	407.426	6642	10148976	405.959	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	331	505768	20.2307
25	6666	10185648	407.426	6534	9983952	399.358	331	505768	20.2307	324	495072	19.8029	331	505768	20.2307	331	505768	20.2307
26	6666	10185648	407.426	6598	10081744	403.27	331	505768	20.2307	322	492016	19.6806	331	505768	20.2307	331	505768	20.2307
27	6666	10185648	407.426	6156	9406368	376.255	331	505768	20.2307	331	505768	20.2307	331	505768	20.2307	330	504240	20.1696
28	6666	10185648	407.426	6629	10129112	405.164	331	505768	20.2307	330	504240	20.1696	331	505768	20.2307	330	504240	20.1696

Table B.23: Interfering 30M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6424	9815872	392.635	497	759416	30.3766	486	742608	29.7043	497	759416	30.3766	496	757888	30.3155
2	6666	10185648	407.426	5828	8905184	356.207	497	759416	30.3766	448	684544	27.3818	497	759416	30.3766	496	757888	30.3155
3	6666	10185648	407.426	6625	10123000	404.92	497	759416	30.3766	489	747192	29.8877	497	759416	30.3766	496	757888	30.3155
4	6666	10185648	407.426	6309	9640152	385.606	497	759416	30.3766	497	759416	30.3766	497	759416	30.3766	494	754832	30.1933
5	6666	10185648	407.426	6586	10063408	402.536	497	759416	30.3766	495	756360	30.2544	497	759416	30.3766	494	754832	30.1933
6	6666	10185648	407.426	6074	9281072	371.243	497	759416	30.3766	493	753304	30.1322	497	759416	30.3766	492	751776	30.071
7	6666	10185648	407.426	6602	10085696	403.514	497	759416	30.3766	474	724272	28.9709	497	759416	30.3766	490	748720	29.9488
8	6666	10185648	407.426	6503	9936584	397.463	497	759416	30.3766	493	753304	30.1322	497	759416	30.3766	491	750248	30.0099
9	6666	10185648	407.426	6502	9935056	397.402	497	759416	30.3766	494	754832	30.1933	497	759416	30.3766	496	757888	30.3155
10	6666	10185648	407.426	6452	9858656	394.346	497	759416	30.3766	494	754832	30.1933	497	759416	30.3766	497	759416	30.3766
11	6666	10185648	407.426	6648	10158144	406.526	497	759416	30.3766	493	753304	30.1322	497	759416	30.3766	497	759416	30.3766
12	6666	10185648	407.426	6568	10035904	401.436	497	759416	30.3766	495	756360	30.2544	497	759416	30.3766	491	750248	30.0099
13	6666	10185648	407.426	5994	9158832	366.353	497	759416	30.3766	495	756360	30.2544	497	759416	30.3766	495	756360	30.2544
14	6666	10185648	407.426	6560	10023680	400.947	497	759416	30.3766	496	757888	30.3155	497	759416	30.3766	497	759416	30.3766
15	6666	10185648	407.426	6481	9902968	396.119	497	759416	30.3766	496	757888	30.3155	497	759416	30.3766	495	756360	30.2544

Table B.24: Interfering 40M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6438	9837264	393.491	663	1013064	40.5226	651	994728	39.7891	663	1013064	40.5226	662	1011536	40.4614
2	6666	10185648	407.426	6342	9690576	387.623	663	1013064	40.5226	660	1008480	40.3392	663	1013064	40.5226	642	980976	39.239
3	6666	10185648	407.426	6452	9858656	394.346	663	1013064	40.5226	658	1005424	40.217	663	1013064	40.5226	660	1008480	40.3392
4	6666	10185648	407.426	6373	9737944	389.518	663	1013064	40.5226	658	1005424	40.217	663	1013064	40.5226	657	1003896	40.1558
5	6666	10185648	407.426	6157	9407896	376.316	663	1013064	40.5226	649	991672	39.6669	663	1013064	40.5226	658	1005424	40.217
6	6666	10185648	407.426	6102	9323856	372.954	663	1013064	40.5226	659	1006952	40.2781	663	1013064	40.5226	662	1011536	40.4614
7	6666	10185648	407.426	5754	8792112	351.684	663	1013064	40.5226	661	1010008	40.4003	663	1013064	40.5226	660	1008480	40.3392
8	6666	10185648	407.426	5730	8755440	350.218	663	1013064	40.5226	656	1002368	40.0947	663	1013064	40.5226	661	1010008	40.4003
9	6666	10185648	407.426	5768	8813504	352.54	663	1013064	40.5226	658	1005424	40.217	663	1013064	40.5226	659	1006952	40.2781
10	6666	10185648	407.426	5254	8028112	321.124	663	1013064	40.5226	658	1005424	40.217	663	1013064	40.5226	650	993200	39.728
11	6666	10185648	407.426	6272	9583616	383.345	663	1013064	40.5226	659	1006952	40.2781	663	1013064	40.5226	651	994728	39.7891
12	6666	10185648	407.426	6524	9968672	398.747	663	1013064	40.5226	655	1000840	40.0336	663	1013064	40.5226	652	996256	39.8502
13	6666	10185648	407.426	6043	9233704	369.348	663	1013064	40.5226	660	1008480	40.3392	663	1013064	40.5226	661	1010008	40.4003
14	6666	10185648	407.426	6461	9872408	394.896	663	1013064	40.5226	662	1011536	40.4614	663	1013064	40.5226	639	976392	39.0557
15	6666	10185648	407.426	6573	10043544	401.742	663	1013064	40.5226	655	1000840	40.0336	663	1013064	40.5226	662	1011536	40.4614

Table B.25: Interfering 50M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	5998	8553744	342.15	829	1266712	50.6685	825	1260600	50.424	829	1266712	50.6685	826	1262128	50.4851
2	6666	10185648	407.426	5735	8763080	350.523	829	1266712	50.6685	823	1257544	50.3018	829	1266712	50.6685	827	1263656	50.5462
3	6666	10185648	407.426	6001	9169528	366.781	829	1266712	50.6685	825	1260600	50.424	829	1266712	50.6685	827	1263656	50.5462
4	6666	10185648	407.426	6015	9190920	367.637	829	1266712	50.6685	828	1265184	50.6074	829	1266712	50.6685	826	1262128	50.4851
5	6666	10185648	407.426	6021	9200088	368.004	829	1266712	50.6685	793	1211704	48.4682	829	1266712	50.6685	820	1252960	50.1184
6	6666	10185648	407.426	6141	9383448	375.338	829	1266712	50.6685	826	1262128	50.4851	829	1266712	50.6685	822	1256016	50.2406
7	6666	10185648	407.426	6192	9461376	378.455	829	1266712	50.6685	818	1249904	49.9962	829	1266712	50.6685	804	1228512	49.1405
8	6666	10185648	407.426	6201	9475128	379.005	829	1266712	50.6685	818	1249904	49.9962	829	1266712	50.6685	827	1263656	50.5462
9	6666	10185648	407.426	6226	9513328	380.533	829	1266712	50.6685	803	1226984	49.0794	829	1266712	50.6685	824	1259072	50.3629
10	6666	10185648	407.426	6274	9586672	383.467	829	1266712	50.6685	819	1251432	50.0573	829	1266712	50.6685	799	1220872	48.8349
11	6666	10185648	407.426	6420	9809760	392.39	829	1266712	50.6685	824	1259072	50.3629	829	1266712	50.6685	802	1225456	49.0182
12	6666	10185648	407.426	6540	9993120	399.725	829	1266712	50.6685	824	1259072	50.3629	829	1266712	50.6685	811	1239208	49.5683
13	6666	10185648	407.426	6599	10083272	403.331	829	1266712	50.6685	825	1260600	50.424	829	1266712	50.6685	823	1257544	50.3018
14	6666	10185648	407.426	6610	10100080	404.003	829	1266712	50.6685	827	1263656	50.5462	829	1266712	50.6685	822	1256016	50.2406
15	6666	10185648	407.426	6659	10174952	406.998	829	1266712	50.6685	823	1257544	50.3018	829	1266712	50.6685	821	1254488	50.1795

Table B.26: Interfering 60M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6350	9702800	388.112	995	1520360	60.8144	971	1483688	59.3475	995	1520360	60.8144	986	1506608	60.2643
2	6666	10185648	407.426	6452	9858656	394.346	995	1520360	60.8144	987	1508136	60.3254	995	1520360	60.8144	966	1476048	59.0419
3	6666	10185648	407.426	6017	9193976	367.759	995	1520360	60.8144	991	1514248	60.5699	995	1520360	60.8144	982	1500496	60.0198
4	6666	10185648	407.426	6170	9427760	377.11	995	1520360	60.8144	994	1518832	60.7533	995	1520360	60.8144	972	1485216	59.4086
5	6666	10185648	407.426	5882	8987696	359.508	995	1520360	60.8144	953	1456184	58.2474	995	1520360	60.8144	979	1495912	59.8365
6	6666	10185648	407.426	5799	8860872	354.435	995	1520360	60.8144	990	1512720	60.5088	995	1520360	60.8144	973	1486744	59.4698
7	6666	10185648	407.426	6520	9962560	398.502	995	1520360	60.8144	991	1514248	60.5699	995	1520360	60.8144	991	1514248	60.5699
8	6666	10185648	407.426	6278	9592784	383.711	995	1520360	60.8144	992	1515776	60.631	995	1520360	60.8144	982	1500496	60.0198
9	6666	10185648	407.426	6007	9178696	367.148	995	1520360	60.8144	993	1517304	60.6922	995	1520360	60.8144	980	1497440	59.8976
10	6666	10185648	407.426	5961	9108408	364.336	995	1520360	60.8144	962	1469936	58.7974	995	1520360	60.8144	970	1482160	59.2864
11	6666	10185648	407.426	6657	10171896	406.876	995	1520360	60.8144	991	1514248	60.5699	995	1520360	60.8144	983	1502024	60.081
12	6666	10185648	407.426	5473	8362744	334.51	995	1520360	60.8144	990	1512720	60.5088	995	1520360	60.8144	975	1489800	59.592
13	6666	10185648	407.426	6086	9299408	371.976	995	1520360	60.8144	993	1517304	60.6922	995	1520360	60.8144	979	1495912	59.8365
14	6666	10185648	407.426	6228	9516384	380.655	995	1520360	60.8144	986	1506608	60.2643	995	1520360	60.8144	991	1514248	60.5699
15	6666	10185648	407.426	6442	9843376	393.735	995	1520360	60.8144	977	1492856	59.7142	995	1520360	60.8144	987	1508136	60.3254

Table B.27: Interfering 70M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	5582	8529296	341.172	1160	1772480	70.8992	1158	1769424	70.777	1160	1772480	70.8992	1151	1758728	70.3491
2	6666	10185648	407.426	5867	8964776	358.591	1160	1772480	70.8992	1159	1770952	70.8381	1160	1772480	70.8992	1151	1758728	70.3491
3	6666	10185648	407.426	6066	9268848	370.754	1160	1772480	70.8992	1150	1757200	70.288	1160	1772480	70.8992	1149	1755672	70.2269
4	6666	10185648	407.426	6114	9342192	373.688	1160	1772480	70.8992	1136	1735808	69.4323	1160	1772480	70.8992	1149	1755672	70.2269
5	6666	10185648	407.426	6175	9435400	377.416	1160	1772480	70.8992	1154	1763312	70.5325	1160	1772480	70.8992	1154	1763312	70.5325
6	6666	10185648	407.426	6179	9441512	377.66	1160	1772480	70.8992	1118	1708304	68.3322	1160	1772480	70.8992	1159	1770952	70.8381
7	6666	10185648	407.426	6248	9546944	381.878	1160	1772480	70.8992	1157	1767896	70.7158	1160	1772480	70.8992	1154	1763312	70.5325
8	6666	10185648	407.426	6262	9568336	382.733	1160	1772480	70.8992	1157	1767896	70.7158	1160	1772480	70.8992	1147	1752616	70.1046
9	6666	10185648	407.426	6264	9571392	382.856	1160	1772480	70.8992	1151	1758728	70.3491	1160	1772480	70.8992	1143	1746504	69.8602
10	6666	10185648	407.426	6268	9577504	383.1	1160	1772480	70.8992	1145	1749560	69.9824	1160	1772480	70.8992	1153	1761784	70.4714
11	6666	10185648	407.426	6364	9724192	388.968	1160	1772480	70.8992	1156	1766368	70.6547	1160	1772480	70.8992	1155	1764840	70.5936
12	6666	10185648	407.426	6364	9724192	388.968	1160	1772480	70.8992	1154	1763312	70.5325	1160	1772480	70.8992	1159	1770952	70.8381
13	6666	10185648	407.426	6393	9768504	390.74	1160	1772480	70.8992	1141	1743448	69.7379	1160	1772480	70.8992	1147	1752616	70.1046
14	6666	10185648	407.426	6468	9883104	395.324	1160	1772480	70.8992	1138	1738864	69.5546	1160	1772480	70.8992	1158	1769424	70.777
15	6666	10185648	407.426	6499	9930472	397.219	1160	1772480	70.8992	1149	1755672	70.2269	1160	1772480	70.8992	1155	1764840	70.5936

Table B.28: Interfering 80M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	5595	8549160	341.966	1326	2026128	81.0451	1315	2009320	80.3728	1326	2026128	81.0451	1323	2021544	80.8618
2	6666	10185648	407.426	5763	8805864	352.235	1326	2026128	81.0451	1324	2023072	80.9229	1326	2026128	81.0451	1323	2021544	80.8618
3	6666	10185648	407.426	5805	8870040	354.802	1326	2026128	81.0451	1314	2007792	80.3117	1326	2026128	81.0451	1307	1997096	79.8838
4	6666	10185648	407.426	5875	8977000	359.08	1326	2026128	81.0451	1318	2013904	80.5562	1326	2026128	81.0451	1295	1978760	79.1504
5	6666	10185648	407.426	5900	9015200	360.608	1326	2026128	81.0451	1324	2023072	80.9229	1326	2026128	81.0451	1320	2016960	80.6784
6	6666	10185648	407.426	5982	9140496	365.62	1326	2026128	81.0451	1317	2012376	80.495	1326	2026128	81.0451	1289	1969592	78.7837
7	6666	10185648	407.426	6007	9178696	367.148	1326	2026128	81.0451	1321	2018488	80.7395	1326	2026128	81.0451	1323	2021544	80.8618
8	6666	10185648	407.426	6087	9300936	372.037	1326	2026128	81.0451	1319	2015432	80.6173	1326	2026128	81.0451	1323	2021544	80.8618
9	6666	10185648	407.426	6212	9491936	379.677	1326	2026128	81.0451	1309	2000152	80.0061	1326	2026128	81.0451	1317	2012376	80.495
10	6666	10185648	407.426	6226	9513328	380.533	1326	2026128	81.0451	1310	2001680	80.0672	1326	2026128	81.0451	1325	2024600	80.984
11	6666	10185648	407.426	6228	9516384	380.655	1326	2026128	81.0451	1324	2023072	80.9229	1326	2026128	81.0451	1304	1992512	79.7005
12	6666	10185648	407.426	6228	9516384	380.655	1326	2026128	81.0451	1325	2024600	80.984	1326	2026128	81.0451	1310	2001680	80.0672
13	6666	10185648	407.426	6243	9539304	381.572	1326	2026128	81.0451	1323	2021544	80.8618	1326	2026128	81.0451	1311	2003208	80.1283
14	6666	10185648	407.426	6292	9614176	384.567	1326	2026128	81.0451	1323	2021544	80.8618	1326	2026128	81.0451	1313	2006264	80.2506
15	6666	10185648	407.426	6409	9792952	391.718	1326	2026128	81.0451	1315	2009320	80.3728	1326	2026128	81.0451	1324	2023072	80.9229

Table B.29: Interfering 90M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	4952	7566656	302.666	1492	2279776	91.191	1468	2243104	89.7242	1492	2279776	91.191	1492	2279776	91.191
2	6666	10185648	407.426	6571	10040488	401.62	1492	2279776	91.191	1486	2270608	90.8243	1492	2279776	91.191	1489	2275192	91.0077
3	6666	10185648	407.426	5360	8190080	327.603	1492	2279776	91.191	1486	2270608	90.8243	1492	2279776	91.191	1449	2214072	88.5629
4	6666	10185648	407.426	5615	8579720	343.189	1492	2279776	91.191	1476	2255328	90.2131	1492	2279776	91.191	1480	2261440	90.4576
5	6666	10185648	407.426	6444	9846432	393.857	1492	2279776	91.191	1461	2232408	89.2963	1492	2279776	91.191	1490	2276720	91.0688
6	6666	10185648	407.426	6470	9886160	395.446	1492	2279776	91.191	1489	2275192	91.0077	1492	2279776	91.191	1478	2258384	90.3354
7	6666	10185648	407.426	6565	10031320	401.253	1492	2279776	91.191	1488	2273664	90.9466	1492	2279776	91.191	1484	2267552	90.7021
8	6666	10185648	407.426	4741	7244248	289.77	1492	2279776	91.191	1489	2275192	91.0077	1492	2279776	91.191	1476	2255328	90.2131
9	6666	10185648	407.426	5672	8666816	346.673	1492	2279776	91.191	1453	2220184	88.8074	1492	2279776	91.191	1443	2204904	88.1962
10	6666	10185648	407.426	5874	8975472	359.019	1492	2279776	91.191	1483	2266024	90.641	1492	2279776	91.191	1485	2269080	90.7632
11	6666	10185648	407.426	6177	9438456	377.538	1492	2279776	91.191	1491	2278248	91.1299	1492	2279776	91.191	1482	2264496	90.5798
12	6666	10185648	407.426	6153	9401784	376.071	1492	2279776	91.191	1489	2275192	91.0077	1492	2279776	91.191	1483	2266024	90.641
13	6666	10185648	407.426	6432	9828096	393.124	1492	2279776	91.191	1486	2270608	90.8243	1492	2279776	91.191	1482	2264496	90.5798
14	6666	10185648	407.426	5587	8536936	341.477	1492	2279776	91.191	1459	2229352	89.1741	1492	2279776	91.191	1491	2278248	91.1299
15	6666	10185648	407.426	6570	10038960	401.558	1492	2279776	91.191	1481	2262968	90.5187	1492	2279776	91.191	1476	2255328	90.2131

Table B.30: Interfering 100M

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	5689	8692792	347.712	1658	2533424	101.337	1619	2473832	98.9533	1658	2533424	101.337	1613	2464664	98.5866
2	6666	10185648	407.426	6206	9482768	379.311	1658	2533424	101.337	1649	2519672	100.787	1658	2533424	101.337	1652	2524256	100.97
3	6666	10185648	407.426	5943	9080904	363.236	1658	2533424	101.337	1653	2525784	101.031	1658	2533424	101.337	1653	2525784	101.031
4	6666	10185648	407.426	6431	9826568	393.063	1658	2533424	101.337	1646	2515088	100.604	1658	2533424	101.337	1642	2508976	100.359
5	6666	10185648	407.426	6618	10112304	404.492	1658	2533424	101.337	1654	2527312	101.092	1658	2533424	101.337	1652	2524256	100.97
6	6666	10185648	407.426	5829	8906712	356.268	1658	2533424	101.337	1651	2522728	100.909	1658	2533424	101.337	1651	2522728	100.909
7	6666	10185648	407.426	6121	9352888	374.116	1658	2533424	101.337	1635	2498280	99.9312	1658	2533424	101.337	1651	2522728	100.909
8	6666	10185648	407.426	5606	8565968	342.639	1658	2533424	101.337	1652	2524256	100.97	1658	2533424	101.337	1654	2527312	101.092
9	6666	10185648	407.426	6171	9429288	377.172	1658	2533424	101.337	1658	2533424	101.337	1658	2533424	101.337	1644	2512032	100.481
10	6666	10185648	407.426	5091	7779048	311.162	1658	2533424	101.337	1647	2516616	100.665	1658	2533424	101.337	1636	2499808	99.9923
11	6666	10185648	407.426	6153	9401784	376.071	1658	2533424	101.337	1625	2483000	99.32	1658	2533424	101.337	1656	2530368	101.215
12	6666	10185648	407.426	5602	8559856	342.394	1658	2533424	101.337	1620	2475360	99.0144	1658	2533424	101.337	1653	2525784	101.031
13	6666	10185648	407.426	6413	9799064	391.963	1658	2533424	101.337	1649	2519672	100.787	1658	2533424	101.337	1653	2525784	101.031
14	6666	10185648	407.426	5656	8642368	345.695	1658	2533424	101.337	1628	2487584	99.5034	1658	2533424	101.337	1657	2531896	101.276
15	6666	10185648	407.426	6343	9692104	387.684	1658	2533424	101.337	1655	2528840	101.154	1658	2533424	101.337	1653	2525784	101.031
16	6666	10185648	407.426	5299	8096872	323.875	1658	2533424	101.337	1656	2530368	101.215	1658	2533424	101.337	1618	2472304	98.8922
17	6666	10185648	407.426	6571	10040488	401.62	1658	2533424	101.337	1651	2522728	100.909	1658	2533424	101.337	1654	2527312	101.092
18	6666	10185648	407.426	6579	10052712	402.108	1658	2533424	101.337	1644	2512032	100.481	1658	2533424	101.337	1621	2476888	99.0755
19	6666	10185648	407.426	6199	9472072	378.883	1658	2533424	101.337	1644	2512032	100.481	1658	2533424	101.337	1653	2525784	101.031
20	6666	10185648	407.426	6020	9198560	367.942	1658	2533424	101.337	1635	2498280	99.9312	1658	2533424	101.337	1652	2524256	100.97
21	6666	10185648	407.426	5339	8157992	326.32	1658	2533424	101.337	1616	2469248	98.7699	1658	2533424	101.337	1633	2495224	99.809
22	6666	10185648	407.426	5884	8990752	359.63	1658	2533424	101.337	1652	2524256	100.97	1658	2533424	101.337	1645	2513560	100.542
23	6666	10185648	407.426	5779	8830312	353.212	1658	2533424	101.337	1654	2527312	101.092	1658	2533424	101.337	1646	2515088	100.604
24	6666	10185648	407.426	4802	7337456	293.498	1658	2533424	101.337	1642	2508976	100.359	1658	2533424	101.337	1621	2476888	99.0755
25	6666	10185648	407.426	5719	8738632	349.545	1658	2533424	101.337	1640	2505920	100.237	1658	2533424	101.337	1655	2528840	101.154
26	6666	10185648	407.426	6067	9270376	370.815	1658	2533424	101.337	1650	2521200	100.848	1658	2533424	101.337	1645	2513560	100.542
27	6666	10185648	407.426	5850	8938800	357.552	1658	2533424	101.337	1642	2508976	100.359	1658	2533424	101.337	1654	2527312	101.092
28	6666	10185648	407.426	6283	9600424	384.017	1658	2533424	101.337	1654	2527312	101.092	1658	2533424	101.337	1438	2197264	87.8906
29	6666	10185648	407.426	6473	9890744	395.63	1658	2533424	101.337	1638	2502864	100.115	1658	2533424	101.337	1655	2528840	101.154
30	6666	10185648	407.426	5580	8526240	341.05	1658	2533424	101.337	1651	2522728	100.909	1658	2533424	101.337	1649	2519672	100.787
31	6666	10185648	407.426	5916	9039648	361.586	1658	2533424	101.337	1634	2496752	99.8701	1658	2533424	101.337	1627	2486056	99.4422
32	6666	10185648	407.426	5790	8847120	353.885	1658	2533424	101.337	1629	2489112	99.5645	1658	2533424	101.337	1657	2531896	101.276
33	6666	10185648	407.426	6012	9186336	367.453	1658	2533424	101.337	1652	2524256	100.97	1658	2533424	101.337	1649	2519672	100.787
34	6666	10185648	407.426	5455	8335240	333.41	1658	2533424	101.337	1647	2516616	100.665	1658	2533424	101.337	1633	2495224	99.809
35	6666	10185648	407.426	6413	9799064	391.963	1658	2533424	101.337	1654	2527312	101.092	1658	2533424	101.337	1656	2530368	101.215
36	6666	10185648	407.426	5103	7797384	311.895	1658	2533424	101.337	1625	2483000	99.32	1658	2533424	101.337	1642	2508976	100.359
37	6666	10185648	407.426	6153	9401784	376.071	1658	2533424	101.337	1655	2528840	101.154	1658	2533424	101.337	1654	2527312	101.092
38	6666	10185648	407.426	4601	7030328	281.213	1658	2533424	101.337	1630	2490640	99.6256	1658	2533424	101.337	1609	2458552	98.3421
39	6666	10185648	407.426	6469	9884632	395.385	1658	2533424	101.337	1623	2479944	99.1978	1658	2533424	101.337	1628	2487584	99.5034
40	6666	10185648	407.426	6313	9646264	385.851	1658	2533424	101.337	1651	2522728	100.909	1658	2533424	101.337	1648	2518144	100.726

Table B.31: Distance 2

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6633	10135224	405.409	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	162	247536	9.90144
4	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
6	6666	10185648	407.426	6638	10142864	405.715	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
7	6666	10185648	407.426	6550	10008400	400.336	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6640	10145920	405.837	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6658	10173424	406.937	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6660	10176480	407.059	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	165	252120	10.0848

Table B.32: Distance 3

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.33: Distance 4

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.34: Distance 5

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.35: Distance 6

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.36: Distance 7

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.37: Distance 8

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.38: Distance 9

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

Table B.39: Distance 10

RNG	Tx_Packets_A	Tx_Bytes_A	TxOffered_A	Rx_Packets_A	Rx_Bytes_A	Throughput_A	Tx_Packets_B	Tx_Bytes_B	TxOffered_B	Rx_Packets_B	Rx_Bytes_B	Throughput_B	Tx_Packets_C	Tx_Bytes_C	TxOffered_C	Rx_Packets_C	Rx_Bytes_C	Throughput_C
1	6666	10185648	407.426	6613	10104664	404.187	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
2	6666	10185648	407.426	6662	10179536	407.181	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
3	6666	10185648	407.426	6646	10155088	406.204	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
4	6666	10185648	407.426	6621	10116888	404.676	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
5	6666	10185648	407.426	6661	10178008	407.12	165	252120	10.0848	163	249064	9.96256	165	252120	10.0848	162	247536	9.90144
6	6666	10185648	407.426	6593	10074104	402.964	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	164	250592	10.0237
7	6666	10185648	407.426	6642	10148976	405.959	165	252120	10.0848	145	221560	8.8624	165	252120	10.0848	165	252120	10.0848
8	6666	10185648	407.426	6584	10060352	402.414	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848
9	6666	10185648	407.426	6650	10161200	406.448	165	252120	10.0848	164	250592	10.0237	165	252120	10.0848	165	252120	10.0848
10	6666	10185648	407.426	6627	10126056	405.042	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848	165	252120	10.0848

I

Code used for simulation

1BSS.cc

```cpp
#include "ns3/command-line.h"
#include "ns3/config.h"
#include "ns3/uinteger.h"
#include "ns3/boolean.h"
#include "ns3/double.h"
#include "ns3/string.h"
#include "ns3/enum.h"
#include "ns3/log.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/spectrum-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/mobility-helper.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/udp-client-server-helper.h"
#include "ns3/packet-sink-helper.h"
#include "ns3/on-off-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/packet-sink.h"
#include "ns3/yans-wifi-channel.h"
#include "ns3/multi-model-spectrum-channel.h"
#include "ns3/wifi-acknowledgment.h"
#include "ns3/rng-seed-manager.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("feito");

int main (int argc, char *argv[])
{
    bool useRts {false};
    bool useExtendedBlockAck {true};
    bool Bsrp {true};
    bool UlOfdma {true};
    bool BeaconJitter {false};
    double simulationTime {10}; //seconds
    double frequency {5}; //whether 2.4, 5 or 6 GHz
    std::string dlAckSeqType {"AGGR-MU-BAR"};
    int mcs {11}; // -1 indicates an unset value
    uint32_t packetNum = 10000;//4294967295u; //number of MaxPackets to be transmitted
    uint32_t payloadSize = 1472; // must fit in the max TX duration when transmitting at MCS 0 over an RU of 26 tones
    double distance = 1.0;
    std::string PacketPerS = "0.00001";  //number of seconds per packet
    std::string phyModel {"Spectrum"};
    AsciiTraceHelper ascii;
    uint8_t nStreams = 4;
```

```
51      CommandLine cmd (__FILE__);
52      cmd.AddValue ("frequency", "Whether working in the 2.4, 5 or 6 GHz band (other values
        gets rejected)", frequency);
53      cmd.AddValue ("simulationTime", "Simulation time in seconds", simulationTime);
54      cmd.AddValue ("useRts", "Enable/disable RTS/CTS", useRts);
55      cmd.AddValue ("useExtendedBlockAck", "Enable/disable use of extended BACK",
        useExtendedBlockAck);
56      cmd.AddValue ("dlAckType", "Ack sequence type for DL OFDMA (NO-OFDMA, ACK-SU-FORMAT,
        MU-BAR, AGGR-MU-BAR)", dlAckSeqType);
57      cmd.AddValue ("mcs", "if set, limit testing to a specific MCS (0-11)", mcs);
58      cmd.AddValue ("payloadSize", "The application payload size in bytes", payloadSize);
59      cmd.AddValue ("distance", "Distance between AP and STA", distance);
60      cmd.AddValue ("phyModel", "PHY model to use when OFDMA is disabled (Yans or Spectrum).
        If OFDMA is enabled then Spectrum is automatically selected", phyModel);
61      cmd.Parse (argc,argv);
62      LogComponentEnable ("feito", LOG_LEVEL_INFO);
63      /*#region RTS,dlAckSeqType, phyModel initiation */
64      if (useRts)
65      {
66          Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
        ("0"));
67          NS_LOG_INFO ("Using RTS/CTS");
68      }
69      else NS_LOG_INFO ("Not using RTS/CTS");
70
71      if (dlAckSeqType == "ACK-SU-FORMAT")
72      {
73          Config::SetDefault ("ns3::WifiDefaultAckManager::DlMuAckSequenceType",
74                              EnumValue (WifiAcknowledgment::DL_MU_BAR_BA_SEQUENCE));
75          NS_LOG_INFO ("Using ACK-SU-FORMAT");
76      }
77      else if (dlAckSeqType == "MU-BAR")
78      {
79          Config::SetDefault ("ns3::WifiDefaultAckManager::DlMuAckSequenceType",
80                              EnumValue (WifiAcknowledgment::DL_MU_TF_MU_BAR));
81          NS_LOG_INFO ("Using MU-BAR-FORMAT");
82      }
83      else if (dlAckSeqType == "AGGR-MU-BAR")
84      {
85          Config::SetDefault ("ns3::WifiDefaultAckManager::DlMuAckSequenceType",
86                              EnumValue (WifiAcknowledgment::DL_MU_AGGREGATE_TF));
87          NS_LOG_INFO ("Using AGGR-MU-FORMAT");
88      }
89      else if (dlAckSeqType != "NO-OFDMA")
90      {
91          NS_ABORT_MSG ("Invalid DL ack sequence type (must be NO-OFDMA, ACK-SU-FORMAT, MU-
        BAR or AGGR-MU-BAR)");
92      }
93      if (phyModel != "Yans" && phyModel != "Spectrum")
94      {
95          NS_ABORT_MSG ("Invalid PHY model (must be Yans or Spectrum)");
96      }
97      if (dlAckSeqType != "NO-OFDMA")
98      {
99          // SpectrumWifiPhy is required for OFDMA
100         phyModel = "Spectrum";
```

```
    }
    /*#endregion*/

    // mudar isto para correr menos MCS
    int minMcs = 9;
    int maxMcs = 11;
    int aux =0;
    if (mcs >= 9 && mcs <= 11)
    {
        minMcs = mcs;
        maxMcs = mcs;
    }
    /*#region logs info*/
    if (frequency ==6)
    {
        NS_LOG_INFO ("Using 6GHZ");
    }
    else if (frequency == 5)
    {
        NS_LOG_INFO ("Using 5GHZ");
    }
    else if (frequency == 2.4)
    {
         NS_LOG_INFO ("Using 2.4GHZ");
    }
    else
    {
        std::cout << "Wrong frequency value!" << std::endl;
        return 0;
    }
    if (UlOfdma)
        NS_LOG_INFO ("Using UlOfdma");
    else
        NS_LOG_INFO ("Not using UlOfdma");
    if (Bsrp)
        NS_LOG_INFO ("Using Bsrp");
    else
        NS_LOG_INFO ("Not using Bsrp");
    if (BeaconJitter)
        NS_LOG_INFO ("Using BeaconJitter");
    else
        NS_LOG_INFO ("Not using BeaconJitter");
    /*#endregion*/
    for (distance = 1 ; distance <=5;){

    for (int mcs = minMcs; mcs <= maxMcs; mcs++)
    {
      uint8_t maxChannelWidth = frequency == 2.4 ? 20 : 160;
      for (int channelWidth = 80 ; channelWidth <= maxChannelWidth;) //MHz
        {
          for (int gi = 3200; gi >= 800; ) //Nanoseconds
            {
                NodeContainer wifiStaNodes;
                wifiStaNodes.Create (1);
                NodeContainer wifiApNode;
```

```
156                 wifiApNode.Create (1);
157                 NetDeviceContainer apDevice, staDevices;
158                 WifiMacHelper mac;
159                 WifiHelper wifi;
160                 if (frequency == 6)
161                 {
162                     wifi.SetStandard (WIFI_STANDARD_80211ax_6GHZ);
163                     Config::SetDefault ("
    ns3::LogDistancePropagationLossModel::ReferenceLoss", DoubleValue (48));
164                 }
165                 else if (frequency == 5)
166                 {
167                     wifi.SetStandard (WIFI_STANDARD_80211ax_5GHZ);
168                 }
169                 else if (frequency == 2.4)
170                 {
171                     wifi.SetStandard (WIFI_STANDARD_80211ax_2_4GHZ);
172                     Config::SetDefault ("
    ns3::LogDistancePropagationLossModel::ReferenceLoss", DoubleValue (40));
173                 }
174 
175                 std::ostringstream oss;
176                 oss << "HeMcs" << mcs;
177                 wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode",
    StringValue (oss.str ()),
178                                               "ControlMode", StringValue (oss.str ()));
179                 Ssid ssid = Ssid ("ns3-80211ax");
180 
181                 Ptr<MultiModelSpectrumChannel> spectrumChannel = CreateObject<
    MultiModelSpectrumChannel> ();
182                 SpectrumWifiPhyHelper phy;
183                 phy.Set ("Antennas", UintegerValue (nStreams));
184                 phy.Set ("MaxSupportedTxSpatialStreams", UintegerValue (nStreams));
185                 phy.Set ("MaxSupportedRxSpatialStreams", UintegerValue (nStreams));
186                 phy.SetChannel (spectrumChannel);
187                 mac.SetType ("ns3::StaWifiMac",
188                              "Ssid", SsidValue (ssid));
189                 phy.Set ("ChannelWidth", UintegerValue (channelWidth));
190                 staDevices = wifi.Install (phy, mac, wifiStaNodes);
191 
192                 if (dlAckSeqType != "NO-OFDMA")
193                 {
194                     mac.SetMultiUserScheduler ("ns3::RrMultiUserScheduler",
195                                                "EnableUlOfdma", BooleanValue (UlOfdma),
196                                                "EnableBsrp", BooleanValue (Bsrp));
197 
198                 }
199                 mac.SetType ("ns3::ApWifiMac",
200                              "EnableBeaconJitter", BooleanValue (BeaconJitter),
201                              "Ssid", SsidValue (ssid));
202                 phy.Set ("ChannelWidth", UintegerValue (channelWidth));
203                 apDevice = wifi.Install (phy, mac, wifiApNode);
204 
205                 RngSeedManager::SetSeed (1);
206                 RngSeedManager::SetRun (1);
207                // int64_t streamNumber = 100;
```

```
208                // streamNumber += wifi.AssignStreams (apDevice, streamNumber);
209                // streamNumber += wifi.AssignStreams (staDevices, streamNumber);
210                 // Set guard interval and MPDU buffer size
211                 Config::Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/HeConfiguration
/GuardInterval", TimeValue (NanoSeconds (gi)));
212                 Config::Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/HeConfiguration
/MpduBufferSize", UintegerValue (useExtendedBlockAck ? 256 : 64));
213                 // mobility.
214                 MobilityHelper mobility;
215                 Ptr<ListPositionAllocator> positionAlloc = CreateObject<
ListPositionAllocator> ();
216
217                 positionAlloc->Add (Vector (0.0, 0.0, 0.0));
218                 positionAlloc->Add (Vector (distance, 0.0, 0.0)); // meter distance
219                 mobility.SetPositionAllocator (positionAlloc);
220
221                 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
222
223                 mobility.Install (wifiApNode);
224                 mobility.Install (wifiStaNodes);
225                 /* Internet stack*/
226                 InternetStackHelper stack;
227                 stack.Install (wifiApNode);
228                 stack.Install (wifiStaNodes);
229
230                 Ipv4AddressHelper address;
231                 address.SetBase ("192.168.1.0", "255.255.255.0");
232                 Ipv4InterfaceContainer staNodeInterfaces;
233                 Ipv4InterfaceContainer apNodeInterface;
234
235                 staNodeInterfaces = address.Assign (staDevices);
236                 apNodeInterface = address.Assign (apDevice);
237
238                 /* Setting applications */
239                 ApplicationContainer serverApp;
240
241                 //UDP flow
242                 uint16_t port = 9;
243                 UdpServerHelper server (port);
244                 serverApp = server.Install (wifiStaNodes);
245                 serverApp.Start (Seconds (0.0));
246                 serverApp.Stop (Seconds (simulationTime + 5));
247
248
249                 UdpClientHelper client (staNodeInterfaces.GetAddress (0), port);
250                 client.SetAttribute ("MaxPackets", UintegerValue (packetNum));
251                 client.SetAttribute ("Interval", TimeValue (Time (PacketPerS)));
//packets/s
252                 client.SetAttribute ("PacketSize", UintegerValue (payloadSize));
253                 ApplicationContainer clientApp = client.Install (wifiApNode.Get (0));
254                 clientApp.Start (Seconds (1.0));
255                 clientApp.Stop (Seconds (simulationTime + 1));
256                 FlowMonitorHelper flowmon;
257                 Ptr<FlowMonitor> monitor = flowmon.InstallAll();
258                 Simulator::Schedule (Seconds (0), &
Ipv4GlobalRoutingHelper::PopulateRoutingTables);
```

```
259                 NS_LOG_UNCOND ("Starting Simulation");
260                 std::string var4 = "Netanim1BSS_dist:"+std::to_string(distance)+"_MCS:"
+std::to_string(mcs)+"_Mhz:" + std::to_string(channelWidth)+"_GI:" + std::to_string(gi)+ "
.xml";
261                 AnimationInterface anim (var4);
262                 phy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);
263                 std::string var2 = "1BSS_AP_dist:"+std::to_string(distance)+"_MCS:"
+std::to_string(mcs)+"_Mhz:" + std::to_string(channelWidth)+"_GI:" + std::to_string(gi)+ "
-pcap";
264                 std::string var3 = "1BSS_STA_dist:"+std::to_string(distance)+"_MCS:"
+std::to_string(mcs)+"_Mhz:" + std::to_string(channelWidth)+"_GI:" + std::to_string(gi)+ "
-pcap";
265                 phy.EnablePcap (var2, apDevice.Get (0), false);
266                 phy.EnablePcap (var3, staDevices.Get (0), false);
267                 //AnimationInterface anim (var4);
268                 //MobilityHelper::EnableAsciiAll (ascii.CreateFileStream (aux + "
teste.mob"));
269                 //phy.EnableAsciiAll(ascii.CreateFileStream(aux+"teste.tr"));
270                 Simulator::Stop (Seconds (simulationTime + 10));
271                 Simulator::Run ();
272
273
274                 /*#beginregion FLOWMONITOR*/
275                 monitor->CheckForLostPackets ();
276                 Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
277                 std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
278
279                 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter =
stats.begin (); iter != stats.end (); ++iter)
280                 {
281                     Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
282                     NS_LOG_UNCOND("Flow_ID " << aux <<
283                     " NUM_Packets "<< packetNum <<" Payload_Size " << payloadSize <<"
PACKETS_PER_S "<< PacketPerS << " MCS "<< mcs << " Width_MHz " << channelWidth << " GI_ns
" << gi << " Distance " << distance <<
284                     " Src_Addr " << t.sourceAddress <<
285                     " Dst_Addr " << t.destinationAddress <<
286                     " Tx_Packets " << iter->second.txPackets <<
287                     " Rx_Packets " << iter->second.rxPackets <<
288                     " Lost_Packets " << iter->second.txPackets-iter->second.rxPackets <<
289                     " Throughput " << iter->second.rxBytes * 8.0 / (iter->
second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1024
<< " Kbps" <<
290                     " Packet_Loss_Ratio " << (iter->second.txPackets-iter->
second.rxPackets)*100/iter->second.txPackets << "%" <<
291                     " Delay " <<iter->second.delaySum << " Jitter " <<iter->
second.jitterSum);
292
293
294                 }
295                 std::string var = "FlowMonitor1BSS_dist:"+std::to_string(distance)+"_MCS:"
+std::to_string(mcs)+"_Mhz:" + std::to_string(channelWidth)+"_GI:" + std::to_string(gi)+ "
.xml";
296                 monitor->SerializeToXmlFile(var, true, true);
297                 /*#endregion*/
298
299                 aux++;
300                 Simulator::Destroy ();
```

```
                gi /= 2;
            }
            channelWidth *= 2;
        }


    }
    distance= distance + 0.5;
    }
    return 0;
}
```

wifi-AP-interference_test3.cc

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2019 University of Washington
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Sébastien Deronne <sebastien.deronne@gmail.com>
 */

//
//  This example program can be used to experiment with spatial
//  reuse mechanisms of 802.11ax.
//
//  The geometry is as follows:
//
//               (STA1)           STA2
//                 |                |
//           d1    |                |d3
//                 |         d2     |-------------STA3
//                (AP1) -----------(AP2)    d4
//
//  STA1 and AP1 are in one BSS (with color set to 1), while STA2 and AP2 are in
//  another BSS (with color set to 2). The distances are configurable (d1 through d3).
//
//  STA1 is continously transmitting data to AP1, while STA2 is continuously sending data to AP2.
//  Each STA has configurable traffic loads (inter packet interval and packet size).
//  It is also possible to configure TX power per node as well as their CCA-ED tresholds.
//  OBSS_PD spatial reuse feature can be enabled (default) or disabled, and the OBSS_PD
//  threshold can be set as well (default: -72 dBm).
//  A simple Friis path loss model is used and a constant PHY rate is considered.
//
//  In general, the program can be configured at run-time by passing command-line arguments.
//  The following command will display all of the available run-time help options:
//    ./waf --run "wifi-spatial-reuse --help"
//
//  By default, the script shows the benefit of the OBSS_PD spatial reuse script:
//    ./waf --run wifi-spatial-reuse
//    Throughput for BSS 1: 6.6468 Mbit/s
```

```
//    Throughput for BSS 2: 6.6672 Mbit/s
//
// If one disables the OBSS_PD feature, a lower throughput is obtained per BSS:
//    ./waf --run "wifi-spatial-reuse --enableObssPd=0"
//    Throughput for BSS 1: 5.8692 Mbit/s
//    Throughput for BSS 2: 5.9364 Mbit/
//
// This difference between those results is because OBSS_PD spatial
// enables to ignore transmissions from another BSS when the received power
// is below the configured threshold, and therefore either defer during ongoing
// transmission or transmit at the same time.
//
#include <string>
#include "ns3/command-line.h"
#include "ns3/config-store-module.h"
#include "ns3/config.h"
#include "ns3/string.h"
#include "ns3/spectrum-wifi-helper.h"
#include "ns3/wifi-acknowledgment.h"
#include "ns3/ssid.h"
#include "ns3/mobility-helper.h"
#include "ns3/application-container.h"
#include "ns3/multi-model-spectrum-channel.h"
#include "ns3/wifi-net-device.h"
#include "ns3/ap-wifi-mac.h"
#include "ns3/he-configuration.h"
#include "ns3/packet-socket-helper.h"
#include "ns3/packet-socket-client.h"
#include "ns3/packet-socket-server.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/udp-echo-helper.h"
#include "ns3/application-container.h"
#include "ns3/on-off-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/ipv4-flow-classifier.h"
#include "ns3/animation-interface.h"
#include "ns3/rng-seed-manager.h"
#include "ns3/he-ru.h"

using namespace ns3;
using namespace std;

std::vector<uint32_t> bytesReceived(5);

uint32_t ContextToNodeId(std::string context) {
    std::string sub = context.substr(10);
    uint32_t pos = sub.find("/Device");
    return atoi(sub.substr(0, pos).c_str());
}

void SocketRx(std::string context, Ptr<const Packet> p, const Address &addr) {
    uint32_t nodeId = ContextToNodeId(context);
    bytesReceived[nodeId] += p->GetSize();
}
```

```
105
106 int main(int argc, char *argv[]) {
107     double duration = 0.2; //10.0; // seconds
108     double d1 = 2.0; // meters
109     double d2 = 10.0; // meters
110     double d3 = 10.0; // meters
111     double d4 = 10.0; // meters
112     double powSta1 = 10.0; // dBm
113     double powSta2 = 10.0; // dBm
114     double powSta3 = 10.0; // dBm
115     double powAp1 = 21.0; // dBm
116     double powAp2 = 21.0; // dBm
117     double ccaEdTrSta1 = -62; // dBm
118     double ccaEdTrSta2 = -62; // dBm
119     double ccaEdTrSta3 = -62; // dBm
120     double ccaEdTrAp1 = -62; // dBm
121     double ccaEdTrAp2 = -62; // dBm
122     uint32_t payloadSize = 1500; // bytes
123     uint32_t payloadSize2 = 1500; // bytes
124     int gi = 800;
125     int RNG = 69;
126     //int Frequency = 5250; // UintegerValue(5210))g - channel 42 at 80 MHz /
    UintegerValue(5250)) - channel 50 at 160 MHz
127     int chwidth =160;
128     //uint32_t CatBytes = 7500000; //bytes
129     //uint32_t IntBytes = 15000; //bytes
130     uint32_t mcs = 0; // MCS value
131     StringValue flowCatDataRate= StringValue("400000000");  // Catheter flow
132     StringValue flowIntDataRate= StringValue("0");  // Interfering flows
133     //string flowIntDataRate2= "0";
134     const double startDataFlows = 1.0; //1.0; // seconds
135     const double stopDataFlows = 0.0001;  // seconds before the end of simulation
136     const int numberOfPings = (::floor(10*(startDataFlows-0.1)));
137     double total_duration= startDataFlows+duration+stopDataFlows;
138
139
140     bool enableObssPd = true;
141     double obssPdThreshold = -72.0; // dBm
142
143     CommandLine cmd(__FILE__);
144     cmd.AddValue("duration", "Duration of simulation (s)", duration);
145     cmd.AddValue("enableObssPd", "Enable/disable OBSS_PD", enableObssPd);
146     cmd.AddValue("d1", "Distance between STA1 and AP1 (m)", d1);
147     cmd.AddValue("d2", "Distance between STA2 and AP2 (m)", d2);
148     cmd.AddValue("d3", "Distance between AP1 and AP2 (m)", d3);
149     cmd.AddValue("d4", "Distance between STA3 and AP2 (m)", d4);
150     cmd.AddValue("powSta1", "Power of STA1 (dBm)", powSta1);
151     cmd.AddValue("powSta2", "Power of STA2 (dBm)", powSta2);
152     cmd.AddValue("powSta2", "Power of STA3 (dBm)", powSta3);
153     cmd.AddValue("powAp1", "Power of AP1 (dBm)", powAp1);
154     cmd.AddValue("powAp2", "Power of AP2 (dBm)", powAp2);
155     cmd.AddValue("ccaEdTrSta1", "CCA-ED Threshold of STA1 (dBm)", ccaEdTrSta1);
156     cmd.AddValue("ccaEdTrSta2", "CCA-ED Threshold of STA2 (dBm)", ccaEdTrSta2);
157     cmd.AddValue("ccaEdTrSta3", "CCA-ED Threshold of STA3 (dBm)", ccaEdTrSta3);
158     cmd.AddValue("ccaEdTrAp1", "CCA-ED Threshold of AP1 (dBm)", ccaEdTrAp1);
```

```
159     cmd.AddValue("mcs", "The constant MCS value to transmit HE PPDUs", mcs);
160     //cmd.AddValue("Frequency", "The Frequency that is used for communication", Frequency)
    ;
161     cmd.AddValue("RNG", "RNG number for seeding", RNG);
162     cmd.AddValue("chwidth","Channel Width", chwidth);
163     cmd.AddValue("gi","Guard Interval in ns", gi);
164     //cmd.AddValue("flowIntDataRate2", "Testing", flowIntDataRate2);
165     cmd.Parse(argc, argv);
166 
167     //flowIntDataRate=flowIntDataRate2;
168 
169     RngSeedManager::SetSeed (RNG);
170     RngSeedManager::SetRun (RNG);
171     Config :: SetDefault ("ns3::WifiDefaultAckManager::DlMuAckSequenceType", EnumValue
    (WifiAcknowledgment::DL_MU_AGGREGATE_TF ));
172     NodeContainer wifiStaNodes;
173     wifiStaNodes.Create(1 + 2);  // 0 - A;  1,2 - B
174 
175     NodeContainer wifiApNodes;
176     wifiApNodes.Create(1 + 1);    // 0 - A;  1 - B
177 
178     SpectrumWifiPhyHelper spectrumPhy;
179     Ptr<MultiModelSpectrumChannel> spectrumChannel = CreateObject<
180             MultiModelSpectrumChannel>();
181     /*Ptr<FriisPropagationLossModel> lossModel = CreateObject<
182             FriisPropagationLossModel>();
183     spectrumChannel->AddPropagationLossModel(lossModel);
184     Ptr<ConstantSpeedPropagationDelayModel> delayModel = CreateObject<
185             ConstantSpeedPropagationDelayModel>();
186     spectrumChannel->SetPropagationDelayModel(delayModel);
187 */
188     spectrumPhy.SetChannel(spectrumChannel);
189     //spectrumPhy.SetErrorRateModel("ns3::YansErrorRateModel");
190 
191     spectrumPhy.Set ("ChannelWidth", UintegerValue (chwidth));
192     spectrumPhy.Set("Antennas", UintegerValue (4));
193     spectrumPhy.Set("MaxSupportedTxSpatialStreams", UintegerValue (4) );
194     spectrumPhy.Set("MaxSupportedRxSpatialStreams", UintegerValue (4) );
195     //spectrumPhy.Set("Frequency", UintegerValue (Frequency));
196 
197     //spectrumPhy.SetPreambleDetectionModel(
198     //      "ns3::ThresholdPreambleDetectionModel");
199     //TODO: add parameter to configure CCA-PD
200 
201     WifiHelper wifi;
202     wifi.SetStandard(WIFI_STANDARD_80211ax_5GHZ);
203     if (enableObssPd) {
204         wifi.SetObssPdAlgorithm("ns3::ConstantObssPdAlgorithm", "ObssPdLevel",
205                 DoubleValue(obssPdThreshold));
206     }
207 
208     WifiMacHelper mac;
209     std::ostringstream oss;
210     oss << "HeMcs" << mcs;
211     wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
```

```
212             StringValue(oss.str()), "ControlMode", StringValue(oss.str()));
213
214     spectrumPhy.Set("TxPowerStart", DoubleValue(powSta1));
215     spectrumPhy.Set("TxPowerEnd", DoubleValue(powSta1));
216     spectrumPhy.Set("CcaEdThreshold", DoubleValue(ccaEdTrSta1));
217     spectrumPhy.Set("RxSensitivity", DoubleValue(-92.0));
218
219     mac . SetMultiUserScheduler ("ns3::RrMultiUserScheduler",
220     "EnableUlOfdma", BooleanValue ( false ),
221     "EnableBsrp", BooleanValue ( true ));
222
223     Ssid ssidA = Ssid("A");
224     mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssidA),
225     "QosSupported", BooleanValue ( true ),
226     "BE_BlockAckThreshold", UintegerValue (2),
227     "ActiveProbing", BooleanValue ( false ));
228     NetDeviceContainer staDeviceA = wifi.Install(spectrumPhy, mac,
229             wifiStaNodes.Get(0));
230
231     spectrumPhy.Set("TxPowerStart", DoubleValue(powAp1));
232     spectrumPhy.Set("TxPowerEnd", DoubleValue(powAp1));
233     spectrumPhy.Set("CcaEdThreshold", DoubleValue(ccaEdTrAp1));
234     spectrumPhy.Set("RxSensitivity", DoubleValue(-92.0));
235
236 /*  mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssidA),
237     "BeaconGeneration", BooleanValue (true),
238     "BeaconInterval", TimeValue ( Seconds (2.499584) ),
239     "EnableBeaconJitter", BooleanValue ( false ),
240     "QosSupported", BooleanValue ( true )
241     );
242 */
243     mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssidA),
244     "BeaconGeneration", BooleanValue (true),
245 //  "BeaconInterval", TimeValue ( Seconds (2.499584) ),
246     "EnableBeaconJitter", BooleanValue ( true ),
247     "QosSupported", BooleanValue ( true )
248     );
249     NetDeviceContainer apDeviceAC = wifi.Install(spectrumPhy, mac,
250             wifiApNodes.Get(0));
251
252     Ptr<WifiNetDevice> apDeviceA =
253             apDeviceAC.Get(0)->GetObject<WifiNetDevice>();
254     Ptr<ApWifiMac> apWifiMac = apDeviceA->GetMac()->GetObject<ApWifiMac>();
255     if (enableObssPd) {
256         apDeviceA->GetHeConfiguration()->SetAttribute("BssColor",
257                 UintegerValue(1));
258     }
259
260     spectrumPhy.Set("TxPowerStart", DoubleValue(powSta2));
261     spectrumPhy.Set("TxPowerEnd", DoubleValue(powSta2));
262     spectrumPhy.Set("CcaEdThreshold", DoubleValue(ccaEdTrSta2));
263     spectrumPhy.Set("RxSensitivity", DoubleValue(-92.0));
264
265     Ssid ssidB = Ssid("B");
266     mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssidB));
```

```
267     NetDeviceContainer staDeviceB1 = wifi.Install(spectrumPhy, mac,
268             wifiStaNodes.Get(1));
269     NetDeviceContainer staDeviceB2 = wifi.Install(spectrumPhy, mac,
270             wifiStaNodes.Get(2));
271
272     spectrumPhy.Set("TxPowerStart", DoubleValue(powAp2));
273     spectrumPhy.Set("TxPowerEnd", DoubleValue(powAp2));
274     spectrumPhy.Set("CcaEdThreshold", DoubleValue(ccaEdTrAp2));
275     spectrumPhy.Set("RxSensitivity", DoubleValue(-92.0));
276
277 /*  mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssidB),
278     "BeaconGeneration", BooleanValue (true),
279     "BeaconInterval", TimeValue ( Seconds (2.499584) ),
280     "EnableBeaconJitter", BooleanValue ( false ),
281     "QosSupported", BooleanValue ( true ));
282 */
283     mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssidB),
284     "BeaconGeneration", BooleanValue (true),
285 //  "BeaconInterval", TimeValue ( Seconds (2.499584) ),
286     "EnableBeaconJitter", BooleanValue ( true ),
287     "QosSupported", BooleanValue ( true ));
288     NetDeviceContainer apDeviceBC = wifi.Install(spectrumPhy, mac,
289             wifiApNodes.Get(1));
290
291     Ptr<WifiNetDevice> apDeviceB =
292             apDeviceBC.Get(0)->GetObject<WifiNetDevice>();
293     apWifiMac = apDeviceB->GetMac()->GetObject<ApWifiMac>();
294     if (enableObssPd) {
295         apDeviceB->GetHeConfiguration()->SetAttribute("BssColor",
296                 UintegerValue(2));
297     }
298
299     // ????
300     /*RngSeedManager :: SetSeed (1) ;
301     RngSeedManager :: SetRun (1) ;
302     int64_t streamNumber = 100;
303     streamNumber += wifi . AssignStreams ( apDeviceAC , streamNumber );
304     streamNumber += wifi . AssignStreams ( staDeviceB1 , streamNumber );
305     streamNumber += wifi . AssignStreams ( apDeviceBC , streamNumber );
306     streamNumber += wifi . AssignStreams ( staDeviceB1 , streamNumber );
307     streamNumber += wifi . AssignStreams ( staDeviceB2 , streamNumber );*/
308     // delete?
309     Config :: Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/HeConfiguration
    /GuardInterval", TimeValue ( NanoSeconds (gi)));
310     Config :: Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/HeConfiguration
    /MpduBufferSize", UintegerValue (256));
311     MobilityHelper mobility;
312     Ptr<ListPositionAllocator> positionAlloc = CreateObject<
313             ListPositionAllocator>();
314     positionAlloc->Add(Vector(0.0, 0.0, 0.0)); // AP1
315     positionAlloc->Add(Vector(d2, 0.0, 0.0));  // AP2
316     positionAlloc->Add(Vector(0.0, d1, 0.0));  // STA1
317     positionAlloc->Add(Vector(d2, d3, 0.0));   // STA2
318     positionAlloc->Add(Vector(d2+d4, 0.0, 0.0));   // STA3
319     mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
```

```
320     mobility.SetPositionAllocator(positionAlloc);
321     mobility.Install(wifiApNodes);
322     mobility.Install(wifiStaNodes);
323
324     /////////////////////////////////////////////////////////////////////////////
325     /////////////////////////////////////////////////////////////////////////////
326     /////////////////////////////////////////////////////////////////////////////
327
328     // Install TCP/IP stack & assign IP addresses
329     InternetStackHelper internet;
330     internet.Install(wifiApNodes);
331     internet.Install(wifiStaNodes);
332     Ipv4AddressHelper ipv4A;
333     ipv4A.SetBase("10.0.0.0", "255.255.255.0");
334     ipv4A.Assign(apDeviceAC);  // .1
335     ipv4A.Assign(staDeviceA);  // .2
336     Ipv4AddressHelper ipv4B;
337     ipv4B.SetBase("10.1.0.0", "255.255.255.0");
338     ipv4B.Assign(apDeviceBC);  // .1
339     ipv4B.Assign(staDeviceB1); // .2
340     ipv4B.Assign(staDeviceB2); // .3
341
342     // Install applications: two CBR streams each saturating the channel
343     ApplicationContainer cbrAppsA, cbrAppsB;
344     uint16_t cbrPortA = 12345;
345     OnOffHelper onOffHelperA("ns3::UdpSocketFactory",
346             InetSocketAddress(Ipv4Address("10.0.0.1"), cbrPortA));
347     onOffHelperA.SetAttribute("PacketSize", UintegerValue(payloadSize));
348     //onOffHelperA.SetAttribute("MaxBytes", UintegerValue(CatBytes));
349     onOffHelperA.SetAttribute("OnTime",
350             StringValue("ns3::ConstantRandomVariable[Constant=1]"));
351     onOffHelperA.SetAttribute("OffTime",
352             StringValue("ns3::ConstantRandomVariable[Constant=0]"));
353
354     uint16_t cbrPortB = 12346;
355     OnOffHelper onOffHelperB1("ns3::UdpSocketFactory",
356             InetSocketAddress(Ipv4Address("10.1.0.1"), cbrPortB));
357     onOffHelperB1.SetAttribute("PacketSize", UintegerValue(payloadSize2));
358     //onOffHelperB1.SetAttribute("MaxBytes", UintegerValue(IntBytes));
359     onOffHelperB1.SetAttribute("OnTime",
360             StringValue("ns3::ConstantRandomVariable[Constant=1]"));
361     onOffHelperB1.SetAttribute("OffTime",
362             StringValue("ns3::ConstantRandomVariable[Constant=0]"));
363
364     OnOffHelper onOffHelperB2("ns3::UdpSocketFactory",
365             InetSocketAddress(Ipv4Address("10.1.0.1"), cbrPortB));
366     onOffHelperB2.SetAttribute("PacketSize", UintegerValue(payloadSize2));
367     //onOffHelperB2.SetAttribute("MaxBytes", UintegerValue(IntBytes));
368     onOffHelperB2.SetAttribute("OnTime",
369             StringValue("ns3::ConstantRandomVariable[Constant=1]"));
370     onOffHelperB2.SetAttribute("OffTime",
371             StringValue("ns3::ConstantRandomVariable[Constant=0]"));
372
373
374     // flow 1:  station A -> AP A (Catheter flow)
```

```
375     onOffHelperA.SetAttribute("DataRate", flowCatDataRate);
376     onOffHelperA.SetAttribute("StartTime", TimeValue(Seconds(startDataFlows)));
377     onOffHelperA.SetAttribute("StopTime", TimeValue(Seconds(total_duration-stopDataFlows))
    );
378     cbrAppsA.Add(onOffHelperA.Install(wifiStaNodes.Get(0)));
379 
380     // flow 2:  stations B1,B2 -> AP B (Interfering traffic)
381     //\internal
382     // The slightly different start times and data rates are a workaround
383     // for \bugid{388} and \bugid{912}
384     onOffHelperB1.SetAttribute("DataRate", flowIntDataRate);
385     onOffHelperB1.SetAttribute("StartTime", TimeValue(Seconds(startDataFlows+0.001)));
386     onOffHelperB1.SetAttribute("StopTime", TimeValue(Seconds(total_duration-
    stopDataFlows+0.00001)));
387 //  onOffHelperB1.SetAttribute("StopTime", TimeValue(Seconds(total_duration-0.499)));
388     cbrAppsB.Add(onOffHelperB1.Install(wifiStaNodes.Get(1)));
389 
390     onOffHelperB2.SetAttribute("DataRate", flowIntDataRate);
391     onOffHelperB2.SetAttribute("StartTime", TimeValue(Seconds(startDataFlows+0.001)));
392     onOffHelperB2.SetAttribute("StopTime", TimeValue(Seconds(total_duration-
    stopDataFlows+0.00001)));
393 //  onOffHelperB.SetAttribute("StopTime", TimeValue(Seconds(total_duration-0.499)));
394     cbrAppsB.Add(onOffHelperB2.Install(wifiStaNodes.Get(2)));
395 
396 
397      /** \internal
398       * We also use separate UDP applications that will send a single
399       * packet before the CBR flows start.
400       * This is a workaround for the lack of perfect ARP, see \bugid{187}
401       */
402       uint16_t  echoPort = 9;
403       UdpEchoClientHelper echoClientHelperA (Ipv4Address ("10.0.0.1"), echoPort);
404       echoClientHelperA.SetAttribute ("MaxPackets", UintegerValue (numberOfPings));
405       echoClientHelperA.SetAttribute ("Interval", TimeValue (Seconds (0.1)));
406       echoClientHelperA.SetAttribute ("PacketSize", UintegerValue (10));
407       UdpEchoClientHelper echoClientHelperB1 (Ipv4Address ("10.1.0.1"), echoPort);
408       echoClientHelperB1.SetAttribute ("MaxPackets", UintegerValue (numberOfPings));
409       echoClientHelperB1.SetAttribute ("Interval", TimeValue (Seconds (0.1)));
410       echoClientHelperB1.SetAttribute ("PacketSize", UintegerValue (10));
411       UdpEchoClientHelper echoClientHelperB2 (Ipv4Address ("10.1.0.1"), echoPort);
412       echoClientHelperB2.SetAttribute ("MaxPackets", UintegerValue (numberOfPings));
413       echoClientHelperB2.SetAttribute ("Interval", TimeValue (Seconds (0.1)));
414       echoClientHelperB2.SetAttribute ("PacketSize", UintegerValue (10));
415       ApplicationContainer pingApps;
416 
417       // again using different start times to workaround Bug 388 and Bug 912
418       echoClientHelperA.SetAttribute ("StartTime", TimeValue (Seconds (0.001)));
419       pingApps.Add (echoClientHelperA.Install (wifiStaNodes.Get(0)));
420       echoClientHelperB1.SetAttribute ("StartTime", TimeValue (Seconds (0.006)));
421       pingApps.Add (echoClientHelperB1.Install (wifiStaNodes.Get(1)));
422       echoClientHelperB2.SetAttribute ("StartTime", TimeValue (Seconds (0.011)));
423       pingApps.Add (echoClientHelperB2.Install (wifiStaNodes.Get(2)));
424 
425 
426       // Install FlowMonitor on all nodes
427       FlowMonitorHelper flowmon;
```

```
428       Ptr<FlowMonitor> monitor = flowmon.InstallAll ();
429
430
431   /*
432       // Let's set up some ns-2-like ascii traces, using another helper class
433       AsciiTraceHelper ascii;
434       Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("wifi-AP-interference-
      test.tr");
435       spectrumPhy.EnableAsciiAll (stream);
436       internet.EnableAsciiIpv4All (stream);
437
438
439       // pcap captures on the backbone wifi devices
440       spectrumPhy.EnablePcap ("wifi-AP-interference-test-apA", apDeviceA, false);
441       spectrumPhy.EnablePcap ("wifi-AP-interference-test-staA", staDeviceA, false);
442       spectrumPhy.EnablePcap ("wifi-AP-interference-test-apB", apDeviceB, false);
443       spectrumPhy.EnablePcap ("wifi-AP-interference-test-staB1", staDeviceB1, false);
444       spectrumPhy.EnablePcap ("wifi-AP-interference-test-staB2", staDeviceB2, false);
445
446       AnimationInterface anim ("wifi-AP-interference-test.xml");
447   */
448
449
450
451       // Run simulation for 10 seconds
452       //std::cout << "MCS   " << "Frequency" << "\t" << "Duration" << "\t" << "RNG" << "\t"
      << "distanceA" << "\t" << "FlowInt" << "\t" << "FlowCat" <<'\n';
453       //std::cout << mcs << "\t" << Frequency  << "\t" << duration  << "\t" << RNG << "\t"
      << d1 << "\t" << flowIntDataRate << '\t' << flowCatDataRate << '\n';
454       Simulator::Stop (Seconds (total_duration));
455       Simulator::Run ();
456
457       // Print per flow statistics
458       monitor->CheckForLostPackets ();
459       Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
      (flowmon.GetClassifier ());
460       FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats ();
461       std::cout << "Flow_A " << "Tx_Packets_A " << "Tx_Bytes_A " << "TxOffered_A "
462       << "Rx_Packets_A " << "Rx_Bytes_A " << "Throughput_A "
463
464       << "Flow_B " << "Tx_Packets_B " << "Tx_Bytes_B " << "TxOffered_B "
465       << "Rx_Packets_B " << "Rx_Bytes_B " << "Throughput_B "
466
467       << "Flow_C " << "Tx_Packets_C " << "Tx_Bytes_C " << "TxOffered_C "
468       << "Rx_Packets_C " << "Rx_Bytes_C " << "Throughput_C " << "\n";
469
470       for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i
      != stats.end (); ++i)
471         {
472           // first 3 FlowIds are for ECHO apps, we don't want to display them
473           //
474           // Duration for throughput measurement is 9.0 seconds, since
475           //   StartTime of the OnOffApplication is at about "second 1"
476           // and
477           //   Simulator::Stops at "second 10".
478           if (i->first > 3)
```

```
            {
             /* Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
              std::cout << "Flow " << i->first - 3 << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";
              */
              std::cout << i->second.txPackets << " "
              << i->second.txBytes << " "
              << i->second.txBytes * 8.0 / (duration) / 1000 / 1000  << " "
              << i->second.rxPackets << " "
              << i->second.rxBytes << " "
              << i->second.rxBytes * 8.0 / (duration) / 1000 / 1000  << " " ;
              //std::cout << "  Tx Packets: " << i->second.txPackets << "\n";
              //std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
              //std::cout << "  TxOffered:  " << i->second.txBytes * 8.0 / (duration) / 1000 / 1000  << " Mbps\n";
              //std::cout << "  Rx Packets: " << i->second.rxPackets << "\n";
              //std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
              //std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / (duration) / 1000 / 1000  << " Mbps\n";
            }
          else {
              // PINGs received
             /* Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
              std::cout << "Flow " << i->first - 3 << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";
              std::cout << "  Tx Pings: " << i->second.txPackets << "\n";
              std::cout << "  Rx Pings: " << i->second.rxPackets << "\n";
              */
          }


      // Cleanup
      Simulator::Destroy ();
    }



    return 0;
}

```

Solo Simulation.ipynb

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import statistics
import scipy.stats
import plotly.graph_objects as go
import plotly.express as px
```

In [ ]:
```
dfMCS = pd.read_csv('MCS_results.csv')
dfMCS
```

In [ ]:
```
MCS=[0,1,2,3,4,5,6,7,8,9,10,11]

aux=dfMCS.loc[dfMCS['Channel Width'] == 160]
aux160_3200=aux.loc[dfMCS['Guard Interval'] == 3200]
aux160_1600=aux.loc[dfMCS['Guard Interval'] == 1600]
aux160_800=aux.loc[dfMCS['Guard Interval'] == 800]

aux=dfMCS.loc[dfMCS['Channel Width'] == 80]
aux80_3200=aux.loc[dfMCS['Guard Interval'] == 3200]
aux80_1600=aux.loc[dfMCS['Guard Interval'] == 1600]
aux80_800=aux.loc[dfMCS['Guard Interval'] == 800]
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=MCS, y=aux160_3200["Throughput"],mode='markers+text', na
fig.add_trace(go.Scatter(x=MCS, y=aux160_1600["Throughput"],mode='markers+text', na
fig.add_trace(go.Scatter(x=MCS, y=aux160_800["Throughput"],mode='markers+text', nam
fig.add_trace(go.Scatter(x=MCS, y=aux80_3200["Throughput"],mode='markers+text', nam
fig.add_trace(go.Scatter(x=MCS, y=aux80_1600["Throughput"],mode='markers+text', nam
fig.add_trace(go.Scatter(x=MCS, y=aux80_800["Throughput"],mode='markers+text', name

fig.update_layout(xaxis_title="Modulation Coding Scheme",
                  yaxis_title="Throughput (Mbps)")


fig.show()
```

In [ ]:
```
df01 = pd.read_csv('Solo_0.1s.csv')

mean_Rx_Packets01 = sum(df01['Rx Packets']) / len(df01['Rx Packets'])
mean_Rx_Bytes01 = sum(df01['Rx Bytes']) / len(df01['Rx Bytes'])
mean_Throughput01 = sum(df01['Throughput']) / len(df01['Throughput'])


Tx_Packets01 = df01['Tx Packets'][0]
Tx_Bytes01 = df01['Tx Bytes'][0]
Tx_Offered01= df01['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets01,'Tx_Bytes:',Tx_Bytes01, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets01,' mean_Rx_Bytes:',mean_Rx_Bytes01,' mea
df01
```

In [ ]:
```
df02 = pd.read_csv('Solo_0.2s.csv')

mean_Rx_Packets02 = sum(df02['Rx Packets']) / len(df02['Rx Packets'])
mean_Rx_Bytes02 = sum(df02['Rx Bytes']) / len(df02['Rx Bytes'])
mean_Throughput02 = sum(df02['Throughput']) / len(df02['Throughput'])


Tx_Packets02 = df02['Tx Packets'][0]
Tx_Bytes02 = df02['Tx Bytes'][0]
Tx_Offered02= df02['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets02,'Tx_Bytes:',Tx_Bytes02, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets02,' mean_Rx_Bytes:',mean_Rx_Bytes02,' mea
df02
```

In [ ]:
```
df03 = pd.read_csv('Solo_0.3s.csv')

mean_Rx_Packets03 = sum(df03['Rx Packets']) / len(df03['Rx Packets'])
mean_Rx_Bytes03 = sum(df03['Rx Bytes']) / len(df03['Rx Bytes'])
mean_Throughput03 = sum(df03['Throughput']) / len(df03['Throughput'])


Tx_Packets03 = df03['Tx Packets'][0]
Tx_Bytes03 = df03['Tx Bytes'][0]
Tx_Offered03= df03['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets03,'Tx_Bytes:',Tx_Bytes03, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets03,' mean_Rx_Bytes:',mean_Rx_Bytes03,' mea
df03
```

In [ ]:
```
df04 = pd.read_csv('Solo_0.4s.csv')

mean_Rx_Packets04 = sum(df04['Rx Packets']) / len(df04['Rx Packets'])
mean_Rx_Bytes04 = sum(df04['Rx Bytes']) / len(df04['Rx Bytes'])
mean_Throughput04 = sum(df04['Throughput']) / len(df04['Throughput'])


Tx_Packets04 = df04['Tx Packets'][0]
Tx_Bytes04 = df04['Tx Bytes'][0]
Tx_Offered04= df04['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets04,'Tx_Bytes:',Tx_Bytes04, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets04,' mean_Rx_Bytes:',mean_Rx_Bytes04,' mea
df04
```

In [ ]:
```
df05 = pd.read_csv('Solo_0.5s.csv')

mean_Rx_Packets05 = sum(df05['Rx Packets']) / len(df05['Rx Packets'])
mean_Rx_Bytes05 = sum(df05['Rx Bytes']) / len(df05['Rx Bytes'])
mean_Throughput05 = sum(df05['Throughput']) / len(df05['Throughput'])


Tx_Packets05 = df05['Tx Packets'][0]
Tx_Bytes05 = df05['Tx Bytes'][0]
Tx_Offered05= df05['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets05,'Tx_Bytes:',Tx_Bytes05, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets05,' mean_Rx_Bytes:',mean_Rx_Bytes05,' mea
df05
```

In [ ]:
```
df06 = pd.read_csv('Solo_0.6s.csv')

mean_Rx_Packets06 = sum(df06['Rx Packets']) / len(df06['Rx Packets'])
mean_Rx_Bytes06 = sum(df06['Rx Bytes']) / len(df06['Rx Bytes'])
mean_Throughput06 = sum(df06['Throughput']) / len(df06['Throughput'])


Tx_Packets06 = df06['Tx Packets'][0]
Tx_Bytes06 = df06['Tx Bytes'][0]
Tx_Offered06= df06['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets06,'Tx_Bytes:',Tx_Bytes06, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets06,' mean_Rx_Bytes:',mean_Rx_Bytes06,' mea
df06
```

In [ ]:
```
df07 = pd.read_csv('Solo_0.7s.csv')

mean_Rx_Packets07 = sum(df07['Rx Packets']) / len(df07['Rx Packets'])
mean_Rx_Bytes07 = sum(df07['Rx Bytes']) / len(df07['Rx Bytes'])
mean_Throughput07 = sum(df07['Throughput']) / len(df07['Throughput'])


Tx_Packets07 = df07['Tx Packets'][0]
Tx_Bytes07 = df07['Tx Bytes'][0]
Tx_Offered07= df07['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets07,'Tx_Bytes:',Tx_Bytes07, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets07,' mean_Rx_Bytes:',mean_Rx_Bytes07,' mea
df07
```

In [ ]:
```
df08 = pd.read_csv('Solo_0.8s.csv')

mean_Rx_Packets08 = sum(df08['Rx Packets']) / len(df08['Rx Packets'])
mean_Rx_Bytes08 = sum(df08['Rx Bytes']) / len(df08['Rx Bytes'])
mean_Throughput08 = sum(df08['Throughput']) / len(df08['Throughput'])


Tx_Packets08 = df08['Tx Packets'][0]
Tx_Bytes08 = df08['Tx Bytes'][0]
Tx_Offered08= df08['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets08,'Tx_Bytes:',Tx_Bytes08, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets08,' mean_Rx_Bytes:',mean_Rx_Bytes08,' mea
df08
```

In [ ]:
```
df09 = pd.read_csv('Solo_0.9s.csv')

mean_Rx_Packets09 = sum(df09['Rx Packets']) / len(df09['Rx Packets'])
mean_Rx_Bytes09 = sum(df09['Rx Bytes']) / len(df09['Rx Bytes'])
mean_Throughput09 = sum(df09['Throughput']) / len(df09['Throughput'])


Tx_Packets09 = df09['Tx Packets'][0]
Tx_Bytes09 = df09['Tx Bytes'][0]
Tx_Offered09= df09['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets09,'Tx_Bytes:',Tx_Bytes09, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets09,' mean_Rx_Bytes:',mean_Rx_Bytes09,' mea
df09
```

In [ ]:
```
df10 = pd.read_csv('Solo_1.0s.csv')

mean_Rx_Packets10 = sum(df10['Rx Packets']) / len(df10['Rx Packets'])
mean_Rx_Bytes10 = sum(df10['Rx Bytes']) / len(df10['Rx Bytes'])
mean_Throughput10 = sum(df10['Throughput']) / len(df10['Throughput'])


Tx_Packets10 = df10['Tx Packets'][0]
Tx_Bytes10 = df10['Tx Bytes'][0]
Tx_Offered10= df10['Tx Offered'][0]


print ('Tx_Packets:',Tx_Packets10,'Tx_Bytes:',Tx_Bytes10, ' Tx_Offered:',Tx_Offered
       'mean_Rx_Packets:',mean_Rx_Packets10,' mean_Rx_Bytes:',mean_Rx_Bytes10,' mea
df10
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Box(y=np.array(df01['Throughput']), name="0.1s"))
fig.add_trace(go.Box(y=np.array(df02['Throughput']), name="0.2s"))
fig.add_trace(go.Box(y=np.array(df03['Throughput']), name="0.3s"))
fig.add_trace(go.Box(y=np.array(df04['Throughput']), name="0.4s"))
fig.add_trace(go.Box(y=np.array(df05['Throughput']), name="0.5s"))
#fig.add_trace(go.Box(y=np.array(df06['Throughput']), name="0.6s"))
fig.add_trace(go.Box(y=np.array(df07['Throughput']), name="0.7s"))
#fig.add_trace(go.Box(y=np.array(df08['Throughput']), name="0.8s"))
fig.add_trace(go.Box(y=np.array(df09['Throughput']), name="0.9s"))
fig.add_trace(go.Box(y=np.array(df10['Throughput']), name="1.0s"))
#fig = px.line(data, x="Time", y="OD", color="C-source")
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Throughput (Mbps)")
fig.show()
```

In [ ]:

```
Tx_Packets = [df01['Tx Packets'][0],
              df02['Tx Packets'][0],
              df03['Tx Packets'][0],
              df04['Tx Packets'][0],
              df05['Tx Packets'][0],
              #df06['Tx Packets'][0],
              df07['Tx Packets'][0],
              #df08['Tx Packets'][0],
              df09['Tx Packets'][0],
              df10['Tx Packets'][0] ]

Tx_Bytes = [df01['Tx Bytes'][0],
            df02['Tx Bytes'][0],
            df03['Tx Bytes'][0],
            df04['Tx Bytes'][0],
            df05['Tx Bytes'][0],
            #df06['Tx Bytes'][0],
            df07['Tx Bytes'][0],
            #df08['Tx Bytes'][0],
            df09['Tx Bytes'][0],
            df10['Tx Bytes'][0] ]

Tx_Offered= [df01['Tx Offered'][0],
             df02['Tx Offered'][0],
             df03['Tx Offered'][0],
             df04['Tx Offered'][0],
             df05['Tx Offered'][0],
             #df06['Tx Offered'][0],
             df07['Tx Offered'][0],
             #df08['Tx Offered'][0],
             df09['Tx Offered'][0],
             df10['Tx Offered'][0] ]

Tx_Bytes_per01 = [df01['Tx Bytes'][0],
                  df02['Tx Bytes'][0]/2,
                  df03['Tx Bytes'][0]/3,
                  df04['Tx Bytes'][0]/4,
                  df05['Tx Bytes'][0]/5,
                  #df06['Tx Bytes'][0]/6,
                  df07['Tx Bytes'][0]/7,
                  #df08['Tx Bytes'][0]/8,
                  df09['Tx Bytes'][0]/9,
                  df10['Tx Bytes'][0]/10, ]

mean_Rx_Packets_per01 =[mean_Rx_Packets01,
                        mean_Rx_Packets02/2,
                        mean_Rx_Packets03/3,
                        mean_Rx_Packets04/4,
                        mean_Rx_Packets05/5,
                        #mean_Rx_Packets06/6,
                        mean_Rx_Packets07/7,
                        #mean_Rx_Packets08/8,
                        mean_Rx_Packets09/9,
                        mean_Rx_Packets10/10]

mean_Rx_Bytes_per01 =[mean_Rx_Bytes01/1,
                      mean_Rx_Bytes02/2,
                      mean_Rx_Bytes03/3,
                      mean_Rx_Bytes04/4
```

```
                        mean_Rx_Bytes04/4,
                        mean_Rx_Bytes05/5,
                       # mean_Rx_Bytes06/6,
                        mean_Rx_Bytes07/7,
                        #mean_Rx_Bytes08/8,
                        mean_Rx_Bytes09/9,
                        mean_Rx_Bytes10/10]

mean_Rx_Packets =[mean_Rx_Packets01,
                  mean_Rx_Packets02,
                  mean_Rx_Packets03,
                  mean_Rx_Packets04,
                  mean_Rx_Packets05,
                  #mean_Rx_Packets06,
                  mean_Rx_Packets07,
                  #mean_Rx_Packets08,
                  mean_Rx_Packets09,
                  mean_Rx_Packets10]

mean_Rx_Bytes =[mean_Rx_Bytes01,
                mean_Rx_Bytes02,
                mean_Rx_Bytes03,
                mean_Rx_Bytes04,
                mean_Rx_Bytes05,
                #mean_Rx_Bytes06,
                mean_Rx_Bytes07,
                #mean_Rx_Bytes08,
                mean_Rx_Bytes09,
                mean_Rx_Bytes10]

mean_Throughput =[mean_Throughput01,
                  mean_Throughput02,
                  mean_Throughput03,
                  mean_Throughput04,
                  mean_Throughput05,
                  #mean_Throughput06,
                  mean_Throughput07,
                  #mean_Throughput08,
                  mean_Throughput09,
                  mean_Throughput10]

#Sucess_chance_transmit = (mean_Rx_Packets*100)/(Tx_Packets*1010
print ('Tx_Packets:',Tx_Packets,'\n''Tx_Bytes:',Tx_Bytes,'\n''Tx_Offered:',Tx_Offer
       'mean_Rx_Packets:',mean_Rx_Packets,'\n''mean_Rx_Bytes:',mean_Rx_Bytes,'\n''m
      'Tx_Bytes_per01:',Tx_Bytes_per01,'\n''mean_Rx_Packets_per01:',mean_Rx_Packets
```

In [ ]:

```
x=[0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9, 1.0]
MCS= 11
Frequency= 5250
GI= 800
Antennas= 4
PayloadSize= 1500
flowCatDataRate= 400000000
flowIntDataRate= 1
Distance= 2
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=Tx_Packets,mode='lines+markers+text', name='Tx_Pack
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Number of Packets")


fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=Tx_Bytes,mode='lines+markers+text', name='Tx_Bytes'
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Bytes Transmitted")
#fig.update_layout(title_text="Tx Bytes")
fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Packets,mode='lines+markers+text', name='Rx
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Number of Packets")
fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Bytes,mode='lines+markers+text', name='Rx_B
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Bytes Received")
fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Packets_per01,mode='lines+markers+text', na
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Packets Received per 0.1s")
fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Bytes_per01,mode='lines+markers+text', name
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Bytes Received per 0.1s")
fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=Tx_Bytes_per01,mode='lines+markers+text', name='Tx_
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Bytes Transmitted per 0.1s")
fig.show()
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=Tx_Offered, mode='lines', name='Tx Offered'))
fig.add_trace(go.Scatter(x=x, y=mean_Throughput,mode='lines+markers+text', name='Th
#fig.update_traces(textposition='top center')
#fig.update_yaxes( dtick=0.1)
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Throughput (Mbps)")

#fig.update_layout(title_text="Tx Offered vs Throughput")
fig.show()
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=Tx_Packets, mode='lines', name='Tx Packets'))
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Packets, mode='lines', name='Rx Packets'))

fig.update_layout(title_text="Tx and Rx Packets")
fig.show()
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=Tx_Bytes, mode='lines', name='Tx Bytes'))
fig.add_trace(go.Scatter(x=x, y=mean_Rx_Bytes, mode='lines', name='Rx Bytes'))

fig.update_layout(title_text="Tx and Rx Bytes")
fig.show()
```

In [ ]:
```
x2=["0.1", "0.2", "0.3", "0.4", "0.5", "0.7", "0.9", "1.0"]
y2=mean_Throughput
y3=Tx_Offered

fig = plt.figure()
ax = fig.add_subplot(111)

plt.plot(x2,y2, label="Throughput")
plt.plot(x2,y3, label="Tx Offered")

plt.plot(range(len(x2)), y2, 'bo') # Plotting data
plt.xticks(range(len(x2)), x2) # Redefining x-axis labels
plt.ylim(404,408)
for i, v in enumerate(y2):
    ax.text(i, v+0.15, "%.2f" %v, ha="center")
plt.xlabel("Interfering Data Rate (bps)")
plt.ylabel("Node A Throughput (Mbps)")


plt.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=Tx_Offered,mode='lines+markers+text', name='Tx_Offe
fig.update_layout(xaxis_title="Simulation Time (s)",
                  yaxis_title="Data Rate (Mbps)")
fig.update_layout(yaxis_range=[407,408])

fig.show()
```

Interfering Time.ipynb

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import statistics
import scipy.stats
import plotly.graph_objects as go
```

In [ ]:
```
df001 = pd.read_csv('Simulation_A.csv')

mean_Rx_Packets_001A = sum(df001['Rx_Packets_A']) / len(df001['Rx_Packets_A'])
mean_Rx_Bytes_001A = sum(df001['Rx_Bytes_A']) / len(df001['Rx_Bytes_A'])
mean_Throughput_001A = sum(df001['Throughput_A']) / len(df001['Throughput_A'])

Tx_Packets_001A = df001['Tx_Packets_A'][0]
Tx_Bytes_001A = df001['Tx_Bytes_A'][0]
Tx_Offered_001A= df001['TxOffered_A'][0]


mean_Rx_Packets_001B = sum(df001['Rx_Packets_B']) / len(df001['Rx_Packets_B'])
mean_Rx_Bytes_001B = sum(df001['Rx_Bytes_B']) / len(df001['Rx_Bytes_B'])
mean_Throughput_001B = sum(df001['Throughput_B']) / len(df001['Throughput_B'])

mean_Rx_Packets_001C = sum(df001['Rx_Packets_C']) / len(df001['Rx_Packets_C'])
mean_Rx_Bytes_001C = sum(df001['Rx_Bytes_C']) / len(df001['Rx_Bytes_C'])
mean_Throughput_001C = sum(df001['Throughput_C']) / len(df001['Throughput_C'])



Tx_Packets_001B = df001['Tx_Packets_B'][0]
Tx_Bytes_001B = df001['Tx_Bytes_B'][0]
Tx_Offered_001B= df001['TxOffered_B'][0]

Tx_Packets_001C = df001['Tx_Packets_C'][0]
Tx_Bytes_001C = df001['Tx_Bytes_C'][0]
Tx_Offered_001C= df001['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_001A,'Tx_Bytes_A:',Tx_Bytes_001A, ' Tx_Offered_A:
       'mean_Rx_Packets_A:',mean_Rx_Packets_001A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_

print ('Tx_Packets_B:',Tx_Packets_001B,'Tx_Bytes_B:',Tx_Bytes_001B, ' Tx_Offered_B:
       'mean_Rx_Packets_B:',mean_Rx_Packets_001B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_

print ('Tx_Packets_C:',Tx_Packets_001C,'Tx_Bytes_C:',Tx_Bytes_001C, ' Tx_Offered_C:
       'mean_Rx_Packets_C:',mean_Rx_Packets_001C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_

df001
```

In [ ]:

```
df01 = pd.read_csv('SimulationB.csv')

mean_Rx_Packets_01A = sum(df01['Rx_Packets_A']) / len(df01['Rx_Packets_A'])
mean_Rx_Bytes_01A = sum(df01['Rx_Bytes_A']) / len(df01['Rx_Bytes_A'])
mean_Throughput_01A = sum(df01['Throughput_A']) / len(df01['Throughput_A'])

Tx_Packets_01A = df01['Tx_Packets_A'][0]
Tx_Bytes_01A = df01['Tx_Bytes_A'][0]
Tx_Offered_01A= df01['TxOffered_A'][0]


mean_Rx_Packets_01B = sum(df01['Rx_Packets_B']) / len(df01['Rx_Packets_B'])
mean_Rx_Bytes_01B = sum(df01['Rx_Bytes_B']) / len(df01['Rx_Bytes_B'])
mean_Throughput_01B = sum(df01['Throughput_B']) / len(df01['Throughput_B'])

mean_Rx_Packets_01C = sum(df01['Rx_Packets_C']) / len(df01['Rx_Packets_C'])
mean_Rx_Bytes_01C = sum(df01['Rx_Bytes_C']) / len(df01['Rx_Bytes_C'])
mean_Throughput_01C = sum(df01['Throughput_C']) / len(df01['Throughput_C'])



Tx_Packets_01B = df01['Tx_Packets_B'][0]
Tx_Bytes_01B = df01['Tx_Bytes_B'][0]
Tx_Offered_01B= df01['TxOffered_B'][0]

Tx_Packets_01C = df01['Tx_Packets_C'][0]
Tx_Bytes_01C = df01['Tx_Bytes_C'][0]
Tx_Offered_01C= df01['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_01A,'Tx_Bytes_A:',Tx_Bytes_01A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_01A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_0

print ('Tx_Packets_B:',Tx_Packets_01B,'Tx_Bytes_B:',Tx_Bytes_01B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_01B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_0

print ('Tx_Packets_C:',Tx_Packets_01C,'Tx_Bytes_C:',Tx_Bytes_01C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_01C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_0

df01
```

In [ ]:

```
df02 = pd.read_csv('Simulation_C.csv')

mean_Rx_Packets_02A = sum(df02['Rx_Packets_A']) / len(df02['Rx_Packets_A'])
mean_Rx_Bytes_02A = sum(df02['Rx_Bytes_A']) / len(df02['Rx_Bytes_A'])
mean_Throughput_02A = sum(df02['Throughput_A']) / len(df02['Throughput_A'])

Tx_Packets_02A = df02['Tx_Packets_A'][0]
Tx_Bytes_02A = df02['Tx_Bytes_A'][0]
Tx_Offered_02A= df02['TxOffered_A'][0]


mean_Rx_Packets_02B = sum(df02['Rx_Packets_B']) / len(df02['Rx_Packets_B'])
mean_Rx_Bytes_02B = sum(df02['Rx_Bytes_B']) / len(df02['Rx_Bytes_B'])
mean_Throughput_02B = sum(df02['Throughput_B']) / len(df02['Throughput_B'])

mean_Rx_Packets_02C = sum(df02['Rx_Packets_C']) / len(df02['Rx_Packets_C'])
mean_Rx_Bytes_02C = sum(df02['Rx_Bytes_C']) / len(df02['Rx_Bytes_C'])
mean_Throughput_02C = sum(df02['Throughput_C']) / len(df02['Throughput_C'])



Tx_Packets_02B = df02['Tx_Packets_B'][0]
Tx_Bytes_02B = df02['Tx_Bytes_B'][0]
Tx_Offered_02B= df02['TxOffered_B'][0]

Tx_Packets_02C = df02['Tx_Packets_C'][0]
Tx_Bytes_02C = df02['Tx_Bytes_C'][0]
Tx_Offered_02C= df02['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_02A,'Tx_Bytes_A:',Tx_Bytes_02A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_02A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_0

print ('Tx_Packets_B:',Tx_Packets_02B,'Tx_Bytes_B:',Tx_Bytes_02B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_02B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_0

print ('Tx_Packets_C:',Tx_Packets_02C,'Tx_Bytes_C:',Tx_Bytes_02C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_02C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_0

df02
```

In [ ]:

```
df05 = pd.read_csv('Simulation_D.csv')

mean_Rx_Packets_05A = sum(df05['Rx_Packets_A']) / len(df05['Rx_Packets_A'])
mean_Rx_Bytes_05A = sum(df05['Rx_Bytes_A']) / len(df05['Rx_Bytes_A'])
mean_Throughput_05A = sum(df05['Throughput_A']) / len(df05['Throughput_A'])

Tx_Packets_05A = df05['Tx_Packets_A'][0]
Tx_Bytes_05A = df05['Tx_Bytes_A'][0]
Tx_Offered_05A= df05['TxOffered_A'][0]


mean_Rx_Packets_05B = sum(df05['Rx_Packets_B']) / len(df05['Rx_Packets_B'])
mean_Rx_Bytes_05B = sum(df05['Rx_Bytes_B']) / len(df05['Rx_Bytes_B'])
mean_Throughput_05B = sum(df05['Throughput_B']) / len(df05['Throughput_B'])

mean_Rx_Packets_05C = sum(df05['Rx_Packets_C']) / len(df05['Rx_Packets_C'])
mean_Rx_Bytes_05C = sum(df05['Rx_Bytes_C']) / len(df05['Rx_Bytes_C'])
mean_Throughput_05C = sum(df05['Throughput_C']) / len(df05['Throughput_C'])



Tx_Packets_05B = df05['Tx_Packets_B'][0]
Tx_Bytes_05B = df05['Tx_Bytes_B'][0]
Tx_Offered_05B= df05['TxOffered_B'][0]

Tx_Packets_05C = df05['Tx_Packets_C'][0]
Tx_Bytes_05C = df05['Tx_Bytes_C'][0]
Tx_Offered_05C= df05['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_05A,'Tx_Bytes_A:',Tx_Bytes_05A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_05A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_0

print ('Tx_Packets_B:',Tx_Packets_05B,'Tx_Bytes_B:',Tx_Bytes_05B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_05B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_0

print ('Tx_Packets_C:',Tx_Packets_05C,'Tx_Bytes_C:',Tx_Bytes_05C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_05C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_0

df05
```

In [ ]:
```
df1 = pd.read_csv('Simulation_E.csv')

mean_Rx_Packets_1A = sum(df1['Rx_Packets_A']) / len(df1['Rx_Packets_A'])
mean_Rx_Bytes_1A = sum(df1['Rx_Bytes_A']) / len(df1['Rx_Bytes_A'])
mean_Throughput_1A = sum(df1['Throughput_A']) / len(df1['Throughput_A'])

Tx_Packets_1A = df1['Tx_Packets_A'][0]
Tx_Bytes_1A = df1['Tx_Bytes_A'][0]
Tx_Offered_1A= df1['TxOffered_A'][0]


mean_Rx_Packets_1B = sum(df1['Rx_Packets_B']) / len(df1['Rx_Packets_B'])
mean_Rx_Bytes_1B = sum(df1['Rx_Bytes_B']) / len(df1['Rx_Bytes_B'])
mean_Throughput_1B = sum(df1['Throughput_B']) / len(df1['Throughput_B'])

mean_Rx_Packets_1C = sum(df1['Rx_Packets_C']) / len(df1['Rx_Packets_C'])
mean_Rx_Bytes_1C = sum(df1['Rx_Bytes_C']) / len(df1['Rx_Bytes_C'])
mean_Throughput_1C = sum(df1['Throughput_C']) / len(df1['Throughput_C'])



Tx_Packets_1B = df1['Tx_Packets_B'][0]
Tx_Bytes_1B = df1['Tx_Bytes_B'][0]
Tx_Offered_1B= df1['TxOffered_B'][0]

Tx_Packets_1C = df1['Tx_Packets_C'][0]
Tx_Bytes_1C = df1['Tx_Bytes_C'][0]
Tx_Offered_1C= df1['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_1A,'Tx_Bytes_A:',Tx_Bytes_1A, ' Tx_Offered_A:',Tx
       'mean_Rx_Packets_A:',mean_Rx_Packets_1A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_1A

print ('Tx_Packets_B:',Tx_Packets_1B,'Tx_Bytes_B:',Tx_Bytes_1B, ' Tx_Offered_B:',Tx
       'mean_Rx_Packets_B:',mean_Rx_Packets_1B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_1B

print ('Tx_Packets_C:',Tx_Packets_1C,'Tx_Bytes_C:',Tx_Bytes_1C, ' Tx_Offered_C:',Tx
       'mean_Rx_Packets_C:',mean_Rx_Packets_1C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_1C

df1
```

In [ ]:
```
time = [0.01, 0.1, 0.2, 0.5, 1]
MCS= 11
Frequency= 5250
GI= 800
Antennas= 4
PayloadSize= 1500
flowCatDataRate= 400000000
flowIntDataRate= 10000000
Distance= 2
```

In [ ]:
```
fig_A = go.Figure()

fig_A.add_trace(go.Box(y=np.array(df001      ['Throughput_A']), name="0.01"))
fig_A.add_trace(go.Box(y=np.array(df01     ['Throughput_A']), name="0.1"))
fig_A.add_trace(go.Box(y=np.array(df02    ['Throughput_A']), name="0.2"))
fig_A.add_trace(go.Box(y=np.array(df05   ['Throughput_A']), name="0.5"))
fig_A.add_trace(go.Box(y=np.array(df1  ['Throughput_A']), name="1.0"))

fig_A.update_layout(xaxis_title="Simulation Time (s)",
                    yaxis_title="Node A Throughput (Mbps)")

fig_A.show()
```

In [ ]:
```
fig_B = go.Figure()

fig_B.add_trace(go.Box(y=np.array(df001      ['Throughput_B']), name="0.01"))
fig_B.add_trace(go.Box(y=np.array(df01     ['Throughput_B']), name="0.1"))
fig_B.add_trace(go.Box(y=np.array(df02    ['Throughput_B']), name="0.2"))
fig_B.add_trace(go.Box(y=np.array(df05   ['Throughput_B']), name="0.5"))
fig_B.add_trace(go.Box(y=np.array(df1  ['Throughput_B']), name="1.0"))

fig_B.update_layout(xaxis_title="Simulation Time (s)",
                    yaxis_title="Node B Throughput (Mbps)")

fig_B.show()
```

In [ ]:
```
fig_C = go.Figure()

fig_C.add_trace(go.Box(y=np.array(df001      ['Throughput_C']), name="0.01"))
fig_C.add_trace(go.Box(y=np.array(df01     ['Throughput_C']), name="0.1"))
fig_C.add_trace(go.Box(y=np.array(df02    ['Throughput_C']), name="0.2"))
fig_C.add_trace(go.Box(y=np.array(df05   ['Throughput_C']), name="0.5"))
fig_C.add_trace(go.Box(y=np.array(df1  ['Throughput_C']), name="1.0"))

fig_C.update_layout(xaxis_title="Simulation Time (s)",
                    yaxis_title="Node C Throughput (Mbps)")

fig_C.show()
```

Interfering.ipynb

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import statistics
import scipy.stats
import plotly.graph_objects as go
import plotly.express as px
from scipy.interpolate import interp1d
```

In [ ]:
```
dfInt100 = pd.read_csv('Interfering_100.csv')


mean_Rx_Packets_Int100A = sum(dfInt100['Rx_Packets_A']) / len(dfInt100['Rx_Packets_
mean_Rx_Bytes_Int100A = sum(dfInt100['Rx_Bytes_A']) / len(dfInt100['Rx_Bytes_A'])
mean_Throughput_Int100A = sum(dfInt100['Throughput_A']) / len(dfInt100['Throughput_

Tx_Packets_Int100A = dfInt100['Tx_Packets_A'][0]
Tx_Bytes_Int100A = dfInt100['Tx_Bytes_A'][0]
Tx_Offered_Int100A= dfInt100['TxOffered_A'][0]


mean_Rx_Packets_Int100B = sum(dfInt100['Rx_Packets_B']) / len(dfInt100['Rx_Packets_
mean_Rx_Bytes_Int100B = sum(dfInt100['Rx_Bytes_B']) / len(dfInt100['Rx_Bytes_B'])
mean_Throughput_Int100B = sum(dfInt100['Throughput_B']) / len(dfInt100['Throughput_

mean_Rx_Packets_Int100C = sum(dfInt100['Rx_Packets_C']) / len(dfInt100['Rx_Packets_
mean_Rx_Bytes_Int100C = sum(dfInt100['Rx_Bytes_C']) / len(dfInt100['Rx_Bytes_C'])
mean_Throughput_Int100C = sum(dfInt100['Throughput_C']) / len(dfInt100['Throughput_



Tx_Packets_Int100B = dfInt100['Tx_Packets_B'][0]
Tx_Bytes_Int100B = dfInt100['Tx_Bytes_B'][0]
Tx_Offered_Int100B= dfInt100['TxOffered_B'][0]

Tx_Packets_Int100C = dfInt100['Tx_Packets_C'][0]
Tx_Bytes_Int100C = dfInt100['Tx_Bytes_C'][0]
Tx_Offered_Int100C= dfInt100['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int100A,'Tx_Bytes_A:',Tx_Bytes_Int100A, ' Tx_Offe
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int100A,' mean_Rx_Bytes_A:',mean_Rx_Byt

print ('Tx_Packets_B:',Tx_Packets_Int100B,'Tx_Bytes_B:',Tx_Bytes_Int100B, ' Tx_Offe
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int100B,' mean_Rx_Bytes_B:',mean_Rx_Byt

print ('Tx_Packets_C:',Tx_Packets_Int100C,'Tx_Bytes_C:',Tx_Bytes_Int100C, ' Tx_Offe
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int100C,' mean_Rx_Bytes_C:',mean_Rx_Byt

dfInt100
```

In [ ]:

```
dfInt1000 = pd.read_csv('Interfering_1000.csv')


mean_Rx_Packets_Int1000A = sum(dfInt1000['Rx_Packets_A']) / len(dfInt1000['Rx_Packe
mean_Rx_Bytes_Int1000A = sum(dfInt1000['Rx_Bytes_A']) / len(dfInt1000['Rx_Bytes_A']
mean_Throughput_Int1000A = sum(dfInt1000['Throughput_A']) / len(dfInt1000['Throughp

Tx_Packets_Int1000A = dfInt1000['Tx_Packets_A'][0]
Tx_Bytes_Int1000A = dfInt1000['Tx_Bytes_A'][0]
Tx_Offered_Int1000A= dfInt1000['TxOffered_A'][0]


mean_Rx_Packets_Int1000B = sum(dfInt1000['Rx_Packets_B']) / len(dfInt1000['Rx_Packe
mean_Rx_Bytes_Int1000B = sum(dfInt1000['Rx_Bytes_B']) / len(dfInt1000['Rx_Bytes_B']
mean_Throughput_Int1000B = sum(dfInt1000['Throughput_B']) / len(dfInt1000['Throughp

mean_Rx_Packets_Int1000C = sum(dfInt1000['Rx_Packets_C']) / len(dfInt1000['Rx_Packe
mean_Rx_Bytes_Int1000C = sum(dfInt1000['Rx_Bytes_C']) / len(dfInt1000['Rx_Bytes_C']
mean_Throughput_Int1000C = sum(dfInt1000['Throughput_C']) / len(dfInt1000['Throughp



Tx_Packets_Int1000B = dfInt1000['Tx_Packets_B'][0]
Tx_Bytes_Int1000B = dfInt1000['Tx_Bytes_B'][0]
Tx_Offered_Int1000B= dfInt1000['TxOffered_B'][0]

Tx_Packets_Int1000C = dfInt1000['Tx_Packets_C'][0]
Tx_Bytes_Int1000C = dfInt1000['Tx_Bytes_C'][0]
Tx_Offered_Int1000C= dfInt1000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int1000A,'Tx_Bytes_A:',Tx_Bytes_Int1000A, ' Tx_Of
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int1000A,' mean_Rx_Bytes_A:',mean_Rx_By

print ('Tx_Packets_B:',Tx_Packets_Int1000B,'Tx_Bytes_B:',Tx_Bytes_Int1000B, ' Tx_Of
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int1000B,' mean_Rx_Bytes_B:',mean_Rx_By

print ('Tx_Packets_C:',Tx_Packets_Int1000C,'Tx_Bytes_C:',Tx_Bytes_Int1000C, ' Tx_Of
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int1000C,' mean_Rx_Bytes_C:',mean_Rx_By

dfInt1000
```

In [ ]:

```
dfInt10000 = pd.read_csv('Interfering_10000.csv')


mean_Rx_Packets_Int10000A = sum(dfInt10000['Rx_Packets_A']) / len(dfInt10000['Rx_Pa
mean_Rx_Bytes_Int10000A = sum(dfInt10000['Rx_Bytes_A']) / len(dfInt10000['Rx_Bytes_
mean_Throughput_Int10000A = sum(dfInt10000['Throughput_A']) / len(dfInt10000['Throu

Tx_Packets_Int10000A = dfInt10000['Tx_Packets_A'][0]
Tx_Bytes_Int10000A = dfInt10000['Tx_Bytes_A'][0]
Tx_Offered_Int10000A= dfInt10000['TxOffered_A'][0]


mean_Rx_Packets_Int10000B = sum(dfInt10000['Rx_Packets_B']) / len(dfInt10000['Rx_Pa
mean_Rx_Bytes_Int10000B = sum(dfInt10000['Rx_Bytes_B']) / len(dfInt10000['Rx_Bytes_
mean_Throughput_Int10000B = sum(dfInt10000['Throughput_B']) / len(dfInt10000['Throu

mean_Rx_Packets_Int10000C = sum(dfInt10000['Rx_Packets_C']) / len(dfInt10000['Rx_Pa
mean_Rx_Bytes_Int10000C = sum(dfInt10000['Rx_Bytes_C']) / len(dfInt10000['Rx_Bytes_
mean_Throughput_Int10000C = sum(dfInt10000['Throughput_C']) / len(dfInt10000['Throu



Tx_Packets_Int10000B = dfInt10000['Tx_Packets_B'][0]
Tx_Bytes_Int10000B = dfInt10000['Tx_Bytes_B'][0]
Tx_Offered_Int10000B= dfInt10000['TxOffered_B'][0]

Tx_Packets_Int10000C = dfInt10000['Tx_Packets_C'][0]
Tx_Bytes_Int10000C = dfInt10000['Tx_Bytes_C'][0]
Tx_Offered_Int10000C= dfInt10000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int10000A,'Tx_Bytes_A:',Tx_Bytes_Int10000A, ' Tx_
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int10000A,' mean_Rx_Bytes_A:',mean_Rx_B

print ('Tx_Packets_B:',Tx_Packets_Int10000B,'Tx_Bytes_B:',Tx_Bytes_Int10000B, ' Tx_
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int10000B,' mean_Rx_Bytes_B:',mean_Rx_B

print ('Tx_Packets_C:',Tx_Packets_Int10000C,'Tx_Bytes_C:',Tx_Bytes_Int10000C, ' Tx_
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int10000C,' mean_Rx_Bytes_C:',mean_Rx_B

dfInt10000
```

In [ ]:

```
dfInt100000 = pd.read_csv('Interfering_100000.csv')


mean_Rx_Packets_Int100000A = sum(dfInt100000['Rx_Packets_A']) / len(dfInt100000['Rx
mean_Rx_Bytes_Int100000A = sum(dfInt100000['Rx_Bytes_A']) / len(dfInt100000['Rx_Byt
mean_Throughput_Int100000A = sum(dfInt100000['Throughput_A']) / len(dfInt100000['Th

Tx_Packets_Int100000A = dfInt100000['Tx_Packets_A'][0]
Tx_Bytes_Int100000A = dfInt100000['Tx_Bytes_A'][0]
Tx_Offered_Int100000A= dfInt100000['TxOffered_A'][0]


mean_Rx_Packets_Int100000B = sum(dfInt100000['Rx_Packets_B']) / len(dfInt100000['Rx
mean_Rx_Bytes_Int100000B = sum(dfInt100000['Rx_Bytes_B']) / len(dfInt100000['Rx_Byt
mean_Throughput_Int100000B = sum(dfInt100000['Throughput_B']) / len(dfInt100000['Th

mean_Rx_Packets_Int100000C = sum(dfInt100000['Rx_Packets_C']) / len(dfInt100000['Rx
mean_Rx_Bytes_Int100000C = sum(dfInt100000['Rx_Bytes_C']) / len(dfInt100000['Rx_Byt
mean_Throughput_Int100000C = sum(dfInt100000['Throughput_C']) / len(dfInt100000['Th



Tx_Packets_Int100000B = dfInt100000['Tx_Packets_B'][0]
Tx_Bytes_Int100000B = dfInt100000['Tx_Bytes_B'][0]
Tx_Offered_Int100000B= dfInt100000['TxOffered_B'][0]

Tx_Packets_Int100000C = dfInt100000['Tx_Packets_C'][0]
Tx_Bytes_Int100000C = dfInt100000['Tx_Bytes_C'][0]
Tx_Offered_Int100000C= dfInt100000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int100000A,'Tx_Bytes_A:',Tx_Bytes_Int100000A, ' T
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int100000A,' mean_Rx_Bytes_A:',mean_Rx_

print ('Tx_Packets_B:',Tx_Packets_Int100000B,'Tx_Bytes_B:',Tx_Bytes_Int100000B, ' T
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int100000B,' mean_Rx_Bytes_B:',mean_Rx_

print ('Tx_Packets_C:',Tx_Packets_Int100000C,'Tx_Bytes_C:',Tx_Bytes_Int100000C, ' T
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int100000C,' mean_Rx_Bytes_C:',mean_Rx_

dfInt100000
```

In [ ]:

```
dfInt1000000 = pd.read_csv('Interfering_1000000.csv')


mean_Rx_Packets_Int1000000A = sum(dfInt1000000['Rx_Packets_A']) / len(dfInt1000000[
mean_Rx_Bytes_Int1000000A = sum(dfInt1000000['Rx_Bytes_A']) / len(dfInt1000000['Rx_
mean_Throughput_Int1000000A = sum(dfInt1000000['Throughput_A']) / len(dfInt1000000[

Tx_Packets_Int1000000A = dfInt1000000['Tx_Packets_A'][0]
Tx_Bytes_Int1000000A = dfInt1000000['Tx_Bytes_A'][0]
Tx_Offered_Int1000000A= dfInt1000000['TxOffered_A'][0]


mean_Rx_Packets_Int1000000B = sum(dfInt1000000['Rx_Packets_B']) / len(dfInt1000000[
mean_Rx_Bytes_Int1000000B = sum(dfInt1000000['Rx_Bytes_B']) / len(dfInt1000000['Rx_
mean_Throughput_Int1000000B = sum(dfInt1000000['Throughput_B']) / len(dfInt1000000[

mean_Rx_Packets_Int1000000C = sum(dfInt1000000['Rx_Packets_C']) / len(dfInt1000000[
mean_Rx_Bytes_Int1000000C = sum(dfInt1000000['Rx_Bytes_C']) / len(dfInt1000000['Rx_
mean_Throughput_Int1000000C = sum(dfInt1000000['Throughput_C']) / len(dfInt1000000[



Tx_Packets_Int1000000B = dfInt1000000['Tx_Packets_B'][0]
Tx_Bytes_Int1000000B = dfInt1000000['Tx_Bytes_B'][0]
Tx_Offered_Int1000000B= dfInt1000000['TxOffered_B'][0]

Tx_Packets_Int1000000C = dfInt1000000['Tx_Packets_C'][0]
Tx_Bytes_Int1000000C = dfInt1000000['Tx_Bytes_C'][0]
Tx_Offered_Int1000000C= dfInt1000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int1000000A,'Tx_Bytes_A:',Tx_Bytes_Int1000000A, '
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int1000000A,' mean_Rx_Bytes_A:',mean_Rx

print ('Tx_Packets_B:',Tx_Packets_Int1000000B,'Tx_Bytes_B:',Tx_Bytes_Int1000000B, '
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int1000000B,' mean_Rx_Bytes_B:',mean_Rx

print ('Tx_Packets_C:',Tx_Packets_Int1000000C,'Tx_Bytes_C:',Tx_Bytes_Int1000000C, '
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int1000000C,' mean_Rx_Bytes_C:',mean_Rx

dfInt1000000
```

In [ ]:

```
dfInt10000000 = pd.read_csv('Interfering_10000000.csv')


mean_Rx_Packets_Int10000000A = sum(dfInt10000000['Rx_Packets_A']) / len(dfInt100000
mean_Rx_Bytes_Int10000000A = sum(dfInt10000000['Rx_Bytes_A']) / len(dfInt10000000['
mean_Throughput_Int10000000A = sum(dfInt10000000['Throughput_A']) / len(dfInt100000

Tx_Packets_Int10000000A = dfInt10000000['Tx_Packets_A'][0]
Tx_Bytes_Int10000000A = dfInt10000000['Tx_Bytes_A'][0]
Tx_Offered_Int10000000A= dfInt10000000['TxOffered_A'][0]


mean_Rx_Packets_Int10000000B = sum(dfInt10000000['Rx_Packets_B']) / len(dfInt100000
mean_Rx_Bytes_Int10000000B = sum(dfInt10000000['Rx_Bytes_B']) / len(dfInt10000000['
mean_Throughput_Int10000000B = sum(dfInt10000000['Throughput_B']) / len(dfInt100000

mean_Rx_Packets_Int10000000C = sum(dfInt10000000['Rx_Packets_C']) / len(dfInt100000
mean_Rx_Bytes_Int10000000C = sum(dfInt10000000['Rx_Bytes_C']) / len(dfInt10000000['
mean_Throughput_Int10000000C = sum(dfInt10000000['Throughput_C']) / len(dfInt100000



Tx_Packets_Int10000000B = dfInt10000000['Tx_Packets_B'][0]
Tx_Bytes_Int10000000B = dfInt10000000['Tx_Bytes_B'][0]
Tx_Offered_Int10000000B= dfInt10000000['TxOffered_B'][0]

Tx_Packets_Int10000000C = dfInt10000000['Tx_Packets_C'][0]
Tx_Bytes_Int10000000C = dfInt10000000['Tx_Bytes_C'][0]
Tx_Offered_Int10000000C= dfInt10000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int10000000A,'Tx_Bytes_A:',Tx_Bytes_Int10000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int10000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int10000000B,'Tx_Bytes_B:',Tx_Bytes_Int10000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int10000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int10000000C,'Tx_Bytes_C:',Tx_Bytes_Int10000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int10000000C,' mean_Rx_Bytes_C:',mean_R

dfInt10000000
```

In [ ]:

```
dfInt30000000 = pd.read_csv('Interfering_30000000.csv')


mean_Rx_Packets_Int30000000A = sum(dfInt30000000['Rx_Packets_A']) / len(dfInt300000
mean_Rx_Bytes_Int30000000A = sum(dfInt30000000['Rx_Bytes_A']) / len(dfInt30000000['
mean_Throughput_Int30000000A = sum(dfInt30000000['Throughput_A']) / len(dfInt300000

Tx_Packets_Int30000000A = dfInt30000000['Tx_Packets_A'][0]
Tx_Bytes_Int30000000A = dfInt30000000['Tx_Bytes_A'][0]
Tx_Offered_Int30000000A= dfInt30000000['TxOffered_A'][0]


mean_Rx_Packets_Int30000000B = sum(dfInt30000000['Rx_Packets_B']) / len(dfInt300000
mean_Rx_Bytes_Int30000000B = sum(dfInt30000000['Rx_Bytes_B']) / len(dfInt30000000['
mean_Throughput_Int30000000B = sum(dfInt30000000['Throughput_B']) / len(dfInt300000

mean_Rx_Packets_Int30000000C = sum(dfInt30000000['Rx_Packets_C']) / len(dfInt300000
mean_Rx_Bytes_Int30000000C = sum(dfInt30000000['Rx_Bytes_C']) / len(dfInt30000000['
mean_Throughput_Int30000000C = sum(dfInt30000000['Throughput_C']) / len(dfInt300000



Tx_Packets_Int30000000B = dfInt30000000['Tx_Packets_B'][0]
Tx_Bytes_Int30000000B = dfInt30000000['Tx_Bytes_B'][0]
Tx_Offered_Int30000000B= dfInt30000000['TxOffered_B'][0]

Tx_Packets_Int30000000C = dfInt30000000['Tx_Packets_C'][0]
Tx_Bytes_Int30000000C = dfInt30000000['Tx_Bytes_C'][0]
Tx_Offered_Int30000000C= dfInt30000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int30000000A,'Tx_Bytes_A:',Tx_Bytes_Int30000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int30000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int30000000B,'Tx_Bytes_B:',Tx_Bytes_Int30000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int30000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int30000000C,'Tx_Bytes_C:',Tx_Bytes_Int30000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int30000000C,' mean_Rx_Bytes_C:',mean_R

dfInt30000000
```

In [ ]:

```
dfInt40000000 = pd.read_csv('Interfering_40000000.csv')


mean_Rx_Packets_Int40000000A = sum(dfInt40000000['Rx_Packets_A']) / len(dfInt400000
mean_Rx_Bytes_Int40000000A = sum(dfInt40000000['Rx_Bytes_A']) / len(dfInt40000000['
mean_Throughput_Int40000000A = sum(dfInt40000000['Throughput_A']) / len(dfInt400000

Tx_Packets_Int40000000A = dfInt40000000['Tx_Packets_A'][0]
Tx_Bytes_Int40000000A = dfInt40000000['Tx_Bytes_A'][0]
Tx_Offered_Int40000000A= dfInt40000000['TxOffered_A'][0]


mean_Rx_Packets_Int40000000B = sum(dfInt40000000['Rx_Packets_B']) / len(dfInt400000
mean_Rx_Bytes_Int40000000B = sum(dfInt40000000['Rx_Bytes_B']) / len(dfInt40000000['
mean_Throughput_Int40000000B = sum(dfInt40000000['Throughput_B']) / len(dfInt400000

mean_Rx_Packets_Int40000000C = sum(dfInt40000000['Rx_Packets_C']) / len(dfInt400000
mean_Rx_Bytes_Int40000000C = sum(dfInt40000000['Rx_Bytes_C']) / len(dfInt40000000['
mean_Throughput_Int40000000C = sum(dfInt40000000['Throughput_C']) / len(dfInt400000



Tx_Packets_Int40000000B = dfInt40000000['Tx_Packets_B'][0]
Tx_Bytes_Int40000000B = dfInt40000000['Tx_Bytes_B'][0]
Tx_Offered_Int40000000B= dfInt40000000['TxOffered_B'][0]

Tx_Packets_Int40000000C = dfInt40000000['Tx_Packets_C'][0]
Tx_Bytes_Int40000000C = dfInt40000000['Tx_Bytes_C'][0]
Tx_Offered_Int40000000C= dfInt40000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int40000000A,'Tx_Bytes_A:',Tx_Bytes_Int40000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int40000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int40000000B,'Tx_Bytes_B:',Tx_Bytes_Int40000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int40000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int40000000C,'Tx_Bytes_C:',Tx_Bytes_Int40000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int40000000C,' mean_Rx_Bytes_C:',mean_R

dfInt40000000
```

In [ ]:

```
dfInt60000000 = pd.read_csv('Interfering_60000000.csv')


mean_Rx_Packets_Int60000000A = sum(dfInt60000000['Rx_Packets_A']) / len(dfInt600000
mean_Rx_Bytes_Int60000000A = sum(dfInt60000000['Rx_Bytes_A']) / len(dfInt60000000['
mean_Throughput_Int60000000A = sum(dfInt60000000['Throughput_A']) / len(dfInt600000

Tx_Packets_Int60000000A = dfInt60000000['Tx_Packets_A'][0]
Tx_Bytes_Int60000000A = dfInt60000000['Tx_Bytes_A'][0]
Tx_Offered_Int60000000A= dfInt60000000['TxOffered_A'][0]


mean_Rx_Packets_Int60000000B = sum(dfInt60000000['Rx_Packets_B']) / len(dfInt600000
mean_Rx_Bytes_Int60000000B = sum(dfInt60000000['Rx_Bytes_B']) / len(dfInt60000000['
mean_Throughput_Int60000000B = sum(dfInt60000000['Throughput_B']) / len(dfInt600000

mean_Rx_Packets_Int60000000C = sum(dfInt60000000['Rx_Packets_C']) / len(dfInt600000
mean_Rx_Bytes_Int60000000C = sum(dfInt60000000['Rx_Bytes_C']) / len(dfInt60000000['
mean_Throughput_Int60000000C = sum(dfInt60000000['Throughput_C']) / len(dfInt600000



Tx_Packets_Int60000000B = dfInt60000000['Tx_Packets_B'][0]
Tx_Bytes_Int60000000B = dfInt60000000['Tx_Bytes_B'][0]
Tx_Offered_Int60000000B= dfInt60000000['TxOffered_B'][0]

Tx_Packets_Int60000000C = dfInt60000000['Tx_Packets_C'][0]
Tx_Bytes_Int60000000C = dfInt60000000['Tx_Bytes_C'][0]
Tx_Offered_Int60000000C= dfInt60000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int60000000A,'Tx_Bytes_A:',Tx_Bytes_Int60000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int60000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int60000000B,'Tx_Bytes_B:',Tx_Bytes_Int60000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int60000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int60000000C,'Tx_Bytes_C:',Tx_Bytes_Int60000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int60000000C,' mean_Rx_Bytes_C:',mean_R

dfInt60000000
```

In [ ]:

```
dfInt70000000 = pd.read_csv('Interfering_70000000.csv')


mean_Rx_Packets_Int70000000A = sum(dfInt70000000['Rx_Packets_A']) / len(dfInt700000
mean_Rx_Bytes_Int70000000A = sum(dfInt70000000['Rx_Bytes_A']) / len(dfInt70000000['
mean_Throughput_Int70000000A = sum(dfInt70000000['Throughput_A']) / len(dfInt700000

Tx_Packets_Int70000000A = dfInt70000000['Tx_Packets_A'][0]
Tx_Bytes_Int70000000A = dfInt70000000['Tx_Bytes_A'][0]
Tx_Offered_Int70000000A= dfInt70000000['TxOffered_A'][0]


mean_Rx_Packets_Int70000000B = sum(dfInt70000000['Rx_Packets_B']) / len(dfInt700000
mean_Rx_Bytes_Int70000000B = sum(dfInt70000000['Rx_Bytes_B']) / len(dfInt70000000['
mean_Throughput_Int70000000B = sum(dfInt70000000['Throughput_B']) / len(dfInt700000

mean_Rx_Packets_Int70000000C = sum(dfInt70000000['Rx_Packets_C']) / len(dfInt700000
mean_Rx_Bytes_Int70000000C = sum(dfInt70000000['Rx_Bytes_C']) / len(dfInt70000000['
mean_Throughput_Int70000000C = sum(dfInt70000000['Throughput_C']) / len(dfInt700000



Tx_Packets_Int70000000B = dfInt70000000['Tx_Packets_B'][0]
Tx_Bytes_Int70000000B = dfInt70000000['Tx_Bytes_B'][0]
Tx_Offered_Int70000000B= dfInt70000000['TxOffered_B'][0]

Tx_Packets_Int70000000C = dfInt70000000['Tx_Packets_C'][0]
Tx_Bytes_Int70000000C = dfInt70000000['Tx_Bytes_C'][0]
Tx_Offered_Int70000000C= dfInt70000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int70000000A,'Tx_Bytes_A:',Tx_Bytes_Int70000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int70000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int70000000B,'Tx_Bytes_B:',Tx_Bytes_Int70000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int70000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int70000000C,'Tx_Bytes_C:',Tx_Bytes_Int70000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int70000000C,' mean_Rx_Bytes_C:',mean_R

dfInt70000000
```

In [ ]:

```
dfInt80000000 = pd.read_csv('Interfering_80000000.csv')


mean_Rx_Packets_Int80000000A = sum(dfInt80000000['Rx_Packets_A']) / len(dfInt800000
mean_Rx_Bytes_Int80000000A = sum(dfInt80000000['Rx_Bytes_A']) / len(dfInt80000000['
mean_Throughput_Int80000000A = sum(dfInt80000000['Throughput_A']) / len(dfInt800000

Tx_Packets_Int80000000A = dfInt80000000['Tx_Packets_A'][0]
Tx_Bytes_Int80000000A = dfInt80000000['Tx_Bytes_A'][0]
Tx_Offered_Int80000000A= dfInt80000000['TxOffered_A'][0]


mean_Rx_Packets_Int80000000B = sum(dfInt80000000['Rx_Packets_B']) / len(dfInt800000
mean_Rx_Bytes_Int80000000B = sum(dfInt80000000['Rx_Bytes_B']) / len(dfInt80000000['
mean_Throughput_Int80000000B = sum(dfInt80000000['Throughput_B']) / len(dfInt800000

mean_Rx_Packets_Int80000000C = sum(dfInt80000000['Rx_Packets_C']) / len(dfInt800000
mean_Rx_Bytes_Int80000000C = sum(dfInt80000000['Rx_Bytes_C']) / len(dfInt80000000['
mean_Throughput_Int80000000C = sum(dfInt80000000['Throughput_C']) / len(dfInt800000



Tx_Packets_Int80000000B = dfInt80000000['Tx_Packets_B'][0]
Tx_Bytes_Int80000000B = dfInt80000000['Tx_Bytes_B'][0]
Tx_Offered_Int80000000B= dfInt80000000['TxOffered_B'][0]

Tx_Packets_Int80000000C = dfInt80000000['Tx_Packets_C'][0]
Tx_Bytes_Int80000000C = dfInt80000000['Tx_Bytes_C'][0]
Tx_Offered_Int80000000C= dfInt80000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int80000000A,'Tx_Bytes_A:',Tx_Bytes_Int80000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int80000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int80000000B,'Tx_Bytes_B:',Tx_Bytes_Int80000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int80000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int80000000C,'Tx_Bytes_C:',Tx_Bytes_Int80000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int80000000C,' mean_Rx_Bytes_C:',mean_R

dfInt80000000
```

In [ ]:

```
dfInt90000000 = pd.read_csv('Interfering_90000000.csv')


mean_Rx_Packets_Int90000000A = sum(dfInt90000000['Rx_Packets_A']) / len(dfInt900000
mean_Rx_Bytes_Int90000000A = sum(dfInt90000000['Rx_Bytes_A']) / len(dfInt90000000['
mean_Throughput_Int90000000A = sum(dfInt90000000['Throughput_A']) / len(dfInt900000

Tx_Packets_Int90000000A = dfInt90000000['Tx_Packets_A'][0]
Tx_Bytes_Int90000000A = dfInt90000000['Tx_Bytes_A'][0]
Tx_Offered_Int90000000A= dfInt90000000['TxOffered_A'][0]


mean_Rx_Packets_Int90000000B = sum(dfInt90000000['Rx_Packets_B']) / len(dfInt900000
mean_Rx_Bytes_Int90000000B = sum(dfInt90000000['Rx_Bytes_B']) / len(dfInt90000000['
mean_Throughput_Int90000000B = sum(dfInt90000000['Throughput_B']) / len(dfInt900000

mean_Rx_Packets_Int90000000C = sum(dfInt90000000['Rx_Packets_C']) / len(dfInt900000
mean_Rx_Bytes_Int90000000C = sum(dfInt90000000['Rx_Bytes_C']) / len(dfInt90000000['
mean_Throughput_Int90000000C = sum(dfInt90000000['Throughput_C']) / len(dfInt900000



Tx_Packets_Int90000000B = dfInt90000000['Tx_Packets_B'][0]
Tx_Bytes_Int90000000B = dfInt90000000['Tx_Bytes_B'][0]
Tx_Offered_Int90000000B= dfInt90000000['TxOffered_B'][0]

Tx_Packets_Int90000000C = dfInt90000000['Tx_Packets_C'][0]
Tx_Bytes_Int90000000C = dfInt90000000['Tx_Bytes_C'][0]
Tx_Offered_Int90000000C= dfInt90000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int90000000A,'Tx_Bytes_A:',Tx_Bytes_Int90000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int90000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int90000000B,'Tx_Bytes_B:',Tx_Bytes_Int90000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int90000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int90000000C,'Tx_Bytes_C:',Tx_Bytes_Int90000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int90000000C,' mean_Rx_Bytes_C:',mean_R

dfInt90000000
```

In [ ]:

```
dfInt100000000 = pd.read_csv('Interfering_100000000.csv')


mean_Rx_Packets_Int100000000A = sum(dfInt100000000['Rx_Packets_A']) / len(dfInt1000
mean_Rx_Bytes_Int100000000A = sum(dfInt100000000['Rx_Bytes_A']) / len(dfInt10000000
mean_Throughput_Int100000000A = sum(dfInt100000000['Throughput_A']) / len(dfInt1000

Tx_Packets_Int100000000A = dfInt100000000['Tx_Packets_A'][0]
Tx_Bytes_Int100000000A = dfInt100000000['Tx_Bytes_A'][0]
Tx_Offered_Int100000000A= dfInt100000000['TxOffered_A'][0]


mean_Rx_Packets_Int100000000B = sum(dfInt100000000['Rx_Packets_B']) / len(dfInt1000
mean_Rx_Bytes_Int100000000B = sum(dfInt100000000['Rx_Bytes_B']) / len(dfInt10000000
mean_Throughput_Int100000000B = sum(dfInt100000000['Throughput_B']) / len(dfInt1000

mean_Rx_Packets_Int100000000C = sum(dfInt100000000['Rx_Packets_C']) / len(dfInt1000
mean_Rx_Bytes_Int100000000C = sum(dfInt100000000['Rx_Bytes_C']) / len(dfInt10000000
mean_Throughput_Int100000000C = sum(dfInt100000000['Throughput_C']) / len(dfInt1000



Tx_Packets_Int100000000B = dfInt100000000['Tx_Packets_B'][0]
Tx_Bytes_Int100000000B = dfInt100000000['Tx_Bytes_B'][0]
Tx_Offered_Int100000000B= dfInt100000000['TxOffered_B'][0]

Tx_Packets_Int100000000C = dfInt100000000['Tx_Packets_C'][0]
Tx_Bytes_Int100000000C = dfInt100000000['Tx_Bytes_C'][0]
Tx_Offered_Int100000000C= dfInt100000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int100000000A,'Tx_Bytes_A:',Tx_Bytes_Int100000000
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int100000000A,' mean_Rx_Bytes_A:',mean_

print ('Tx_Packets_B:',Tx_Packets_Int100000000B,'Tx_Bytes_B:',Tx_Bytes_Int100000000
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int100000000B,' mean_Rx_Bytes_B:',mean_

print ('Tx_Packets_C:',Tx_Packets_Int100000000C,'Tx_Bytes_C:',Tx_Bytes_Int100000000
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int100000000C,' mean_Rx_Bytes_C:',mean_

dfInt100000000
```

In [ ]:

```
dfInt2000000 = pd.read_csv('Interfering_2000000.csv')


mean_Rx_Packets_Int2000000A = sum(dfInt2000000['Rx_Packets_A']) / len(dfInt2000000[
mean_Rx_Bytes_Int2000000A = sum(dfInt2000000['Rx_Bytes_A']) / len(dfInt2000000['Rx_
mean_Throughput_Int2000000A = sum(dfInt2000000['Throughput_A']) / len(dfInt2000000[

Tx_Packets_Int2000000A = dfInt2000000['Tx_Packets_A'][0]
Tx_Bytes_Int2000000A = dfInt2000000['Tx_Bytes_A'][0]
Tx_Offered_Int2000000A= dfInt2000000['TxOffered_A'][0]


mean_Rx_Packets_Int2000000B = sum(dfInt2000000['Rx_Packets_B']) / len(dfInt2000000[
mean_Rx_Bytes_Int2000000B = sum(dfInt2000000['Rx_Bytes_B']) / len(dfInt2000000['Rx_
mean_Throughput_Int2000000B = sum(dfInt2000000['Throughput_B']) / len(dfInt2000000[

mean_Rx_Packets_Int2000000C = sum(dfInt2000000['Rx_Packets_C']) / len(dfInt2000000[
mean_Rx_Bytes_Int2000000C = sum(dfInt2000000['Rx_Bytes_C']) / len(dfInt2000000['Rx_
mean_Throughput_Int2000000C = sum(dfInt2000000['Throughput_C']) / len(dfInt2000000[



Tx_Packets_Int2000000B = dfInt2000000['Tx_Packets_B'][0]
Tx_Bytes_Int2000000B = dfInt2000000['Tx_Bytes_B'][0]
Tx_Offered_Int2000000B= dfInt2000000['TxOffered_B'][0]

Tx_Packets_Int2000000C = dfInt2000000['Tx_Packets_C'][0]
Tx_Bytes_Int2000000C = dfInt2000000['Tx_Bytes_C'][0]
Tx_Offered_Int2000000C= dfInt2000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int2000000A,'Tx_Bytes_A:',Tx_Bytes_Int2000000A, '
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int2000000A,' mean_Rx_Bytes_A:',mean_Rx

print ('Tx_Packets_B:',Tx_Packets_Int2000000B,'Tx_Bytes_B:',Tx_Bytes_Int2000000B, '
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int2000000B,' mean_Rx_Bytes_B:',mean_Rx

print ('Tx_Packets_C:',Tx_Packets_Int2000000C,'Tx_Bytes_C:',Tx_Bytes_Int2000000C, '
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int2000000C,' mean_Rx_Bytes_C:',mean_Rx

dfInt2000000
```

In [ ]:

```
dfInt20000000 = pd.read_csv('Interfering_20000000.csv')


mean_Rx_Packets_Int20000000A = sum(dfInt20000000['Rx_Packets_A']) / len(dfInt200000
mean_Rx_Bytes_Int20000000A = sum(dfInt20000000['Rx_Bytes_A']) / len(dfInt20000000['
mean_Throughput_Int20000000A = sum(dfInt20000000['Throughput_A']) / len(dfInt200000

Tx_Packets_Int20000000A = dfInt20000000['Tx_Packets_A'][0]
Tx_Bytes_Int20000000A = dfInt20000000['Tx_Bytes_A'][0]
Tx_Offered_Int20000000A= dfInt20000000['TxOffered_A'][0]


mean_Rx_Packets_Int20000000B = sum(dfInt20000000['Rx_Packets_B']) / len(dfInt200000
mean_Rx_Bytes_Int20000000B = sum(dfInt20000000['Rx_Bytes_B']) / len(dfInt20000000['
mean_Throughput_Int20000000B = sum(dfInt20000000['Throughput_B']) / len(dfInt200000

mean_Rx_Packets_Int20000000C = sum(dfInt20000000['Rx_Packets_C']) / len(dfInt200000
mean_Rx_Bytes_Int20000000C = sum(dfInt20000000['Rx_Bytes_C']) / len(dfInt20000000['
mean_Throughput_Int20000000C = sum(dfInt20000000['Throughput_C']) / len(dfInt200000



Tx_Packets_Int20000000B = dfInt20000000['Tx_Packets_B'][0]
Tx_Bytes_Int20000000B = dfInt20000000['Tx_Bytes_B'][0]
Tx_Offered_Int20000000B= dfInt20000000['TxOffered_B'][0]

Tx_Packets_Int20000000C = dfInt20000000['Tx_Packets_C'][0]
Tx_Bytes_Int20000000C = dfInt20000000['Tx_Bytes_C'][0]
Tx_Offered_Int20000000C= dfInt20000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int20000000A,'Tx_Bytes_A:',Tx_Bytes_Int20000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int20000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int20000000B,'Tx_Bytes_B:',Tx_Bytes_Int20000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int20000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int20000000C,'Tx_Bytes_C:',Tx_Bytes_Int20000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int20000000C,' mean_Rx_Bytes_C:',mean_R

dfInt20000000
```

In [ ]:

```
dfInt5000000 = pd.read_csv('Interfering_5000000.csv')


mean_Rx_Packets_Int5000000A = sum(dfInt5000000['Rx_Packets_A']) / len(dfInt5000000[
mean_Rx_Bytes_Int5000000A = sum(dfInt5000000['Rx_Bytes_A']) / len(dfInt5000000['Rx_
mean_Throughput_Int5000000A = sum(dfInt5000000['Throughput_A']) / len(dfInt5000000[

Tx_Packets_Int5000000A = dfInt5000000['Tx_Packets_A'][0]
Tx_Bytes_Int5000000A = dfInt5000000['Tx_Bytes_A'][0]
Tx_Offered_Int5000000A= dfInt5000000['TxOffered_A'][0]


mean_Rx_Packets_Int5000000B = sum(dfInt5000000['Rx_Packets_B']) / len(dfInt5000000[
mean_Rx_Bytes_Int5000000B = sum(dfInt5000000['Rx_Bytes_B']) / len(dfInt5000000['Rx_
mean_Throughput_Int5000000B = sum(dfInt5000000['Throughput_B']) / len(dfInt5000000[

mean_Rx_Packets_Int5000000C = sum(dfInt5000000['Rx_Packets_C']) / len(dfInt5000000[
mean_Rx_Bytes_Int5000000C = sum(dfInt5000000['Rx_Bytes_C']) / len(dfInt5000000['Rx_
mean_Throughput_Int5000000C = sum(dfInt5000000['Throughput_C']) / len(dfInt5000000[



Tx_Packets_Int5000000B = dfInt5000000['Tx_Packets_B'][0]
Tx_Bytes_Int5000000B = dfInt5000000['Tx_Bytes_B'][0]
Tx_Offered_Int5000000B= dfInt5000000['TxOffered_B'][0]

Tx_Packets_Int5000000C = dfInt5000000['Tx_Packets_C'][0]
Tx_Bytes_Int5000000C = dfInt5000000['Tx_Bytes_C'][0]
Tx_Offered_Int5000000C= dfInt5000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int5000000A,'Tx_Bytes_A:',Tx_Bytes_Int5000000A, '
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int5000000A,' mean_Rx_Bytes_A:',mean_Rx

print ('Tx_Packets_B:',Tx_Packets_Int5000000B,'Tx_Bytes_B:',Tx_Bytes_Int5000000B, '
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int5000000B,' mean_Rx_Bytes_B:',mean_Rx

print ('Tx_Packets_C:',Tx_Packets_Int5000000C,'Tx_Bytes_C:',Tx_Bytes_Int5000000C, '
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int5000000C,' mean_Rx_Bytes_C:',mean_Rx

dfInt5000000
```

In [ ]:
```
dfInt50000000 = pd.read_csv('Interfering_50000000.csv')


mean_Rx_Packets_Int50000000A = sum(dfInt50000000['Rx_Packets_A']) / len(dfInt500000
mean_Rx_Bytes_Int50000000A = sum(dfInt50000000['Rx_Bytes_A']) / len(dfInt50000000['
mean_Throughput_Int50000000A = sum(dfInt50000000['Throughput_A']) / len(dfInt500000

Tx_Packets_Int50000000A = dfInt50000000['Tx_Packets_A'][0]
Tx_Bytes_Int50000000A = dfInt50000000['Tx_Bytes_A'][0]
Tx_Offered_Int50000000A= dfInt50000000['TxOffered_A'][0]


mean_Rx_Packets_Int50000000B = sum(dfInt50000000['Rx_Packets_B']) / len(dfInt500000
mean_Rx_Bytes_Int50000000B = sum(dfInt50000000['Rx_Bytes_B']) / len(dfInt50000000['
mean_Throughput_Int50000000B = sum(dfInt50000000['Throughput_B']) / len(dfInt500000

mean_Rx_Packets_Int50000000C = sum(dfInt50000000['Rx_Packets_C']) / len(dfInt500000
mean_Rx_Bytes_Int50000000C = sum(dfInt50000000['Rx_Bytes_C']) / len(dfInt50000000['
mean_Throughput_Int50000000C = sum(dfInt50000000['Throughput_C']) / len(dfInt500000



Tx_Packets_Int50000000B = dfInt50000000['Tx_Packets_B'][0]
Tx_Bytes_Int50000000B = dfInt50000000['Tx_Bytes_B'][0]
Tx_Offered_Int50000000B= dfInt50000000['TxOffered_B'][0]

Tx_Packets_Int50000000C = dfInt50000000['Tx_Packets_C'][0]
Tx_Bytes_Int50000000C = dfInt50000000['Tx_Bytes_C'][0]
Tx_Offered_Int50000000C= dfInt50000000['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_Int50000000A,'Tx_Bytes_A:',Tx_Bytes_Int50000000A,
       'mean_Rx_Packets_A:',mean_Rx_Packets_Int50000000A,' mean_Rx_Bytes_A:',mean_R

print ('Tx_Packets_B:',Tx_Packets_Int50000000B,'Tx_Bytes_B:',Tx_Bytes_Int50000000B,
       'mean_Rx_Packets_B:',mean_Rx_Packets_Int50000000B,' mean_Rx_Bytes_B:',mean_R

print ('Tx_Packets_C:',Tx_Packets_Int50000000C,'Tx_Bytes_C:',Tx_Bytes_Int50000000C,
       'mean_Rx_Packets_C:',mean_Rx_Packets_Int50000000C,' mean_Rx_Bytes_C:',mean_R

dfInt50000000
```

In [ ]:
```
fig_A1 = go.Figure()

fig_A1.add_trace(go.Box(y=np.array(dfInt100      ['Throughput_A']), name="100"))
fig_A1.add_trace(go.Box(y=np.array(dfInt1000     ['Throughput_A']), name="1 000"))
fig_A1.add_trace(go.Box(y=np.array(dfInt10000    ['Throughput_A']), name="10 000"))
fig_A1.add_trace(go.Box(y=np.array(dfInt100000   ['Throughput_A']), name="100 000"))
fig_A1.add_trace(go.Box(y=np.array(dfInt1000000  ['Throughput_A']), name="1M"))
fig_A1.add_trace(go.Box(y=np.array(dfInt2000000  ['Throughput_A']), name="2M"))
fig_A1.add_trace(go.Box(y=np.array(dfInt5000000  ['Throughput_A']), name="5M"))
fig_A1.add_trace(go.Box(y=np.array(dfInt10000000  ['Throughput_A']), name="10M"))
fig_A1.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node A Throughput (Mbps)")
fig_A1.show()
```

In [ ]:
```
fig_A2 = go.Figure()
fig_A2.add_trace(go.Box(y=np.array(dfInt10000000 ['Throughput_A']), name="10M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt20000000 ['Throughput_A']), name="20M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt30000000 ['Throughput_A']), name="30M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt40000000 ['Throughput_A']), name="40M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt50000000 ['Throughput_A']), name="50M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt60000000 ['Throughput_A']), name="60M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt70000000 ['Throughput_A']), name="70M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt80000000 ['Throughput_A']), name="80M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt90000000 ['Throughput_A']), name="90M"))
fig_A2.add_trace(go.Box(y=np.array(dfInt100000000['Throughput_A']), name="100M"))


fig_A2.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node A Throughput (Mbps)")

fig_A2.show()
```

In [ ]:
```
fig_B1 = go.Figure()

fig_B1.add_trace(go.Box(y=np.array(dfInt100      ['Throughput_B']), name="100"))
fig_B1.add_trace(go.Box(y=np.array(dfInt1000     ['Throughput_B']), name="1 000"))
fig_B1.add_trace(go.Box(y=np.array(dfInt10000    ['Throughput_B']), name="10 000"))
fig_B1.add_trace(go.Box(y=np.array(dfInt100000   ['Throughput_B']), name="100 000")
fig_B1.add_trace(go.Box(y=np.array(dfInt1000000  ['Throughput_B']), name="1M"))
fig_B1.add_trace(go.Box(y=np.array(dfInt2000000  ['Throughput_B']), name="2M"))
fig_B1.add_trace(go.Box(y=np.array(dfInt5000000  ['Throughput_B']), name="5M"))
fig_B1.add_trace(go.Box(y=np.array(dfInt10000000  ['Throughput_B']), name="10M"))
fig_B1.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node B Throughput (Mbps)")

fig_B1.show()
```

In [ ]:
```
fig_B2 = go.Figure()
fig_B2.add_trace(go.Box(y=np.array(dfInt10000000 ['Throughput_B']), name="10M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt20000000 ['Throughput_B']), name="20M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt30000000 ['Throughput_B']), name="30M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt40000000 ['Throughput_B']), name="40M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt50000000 ['Throughput_B']), name="50M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt60000000 ['Throughput_B']), name="60M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt70000000 ['Throughput_B']), name="70M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt80000000 ['Throughput_B']), name="80M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt90000000 ['Throughput_B']), name="90M"))
fig_B2.add_trace(go.Box(y=np.array(dfInt100000000['Throughput_B']), name="100M"))
fig_B2.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node B Throughput (Mbps)")

fig_B2.show()
```

In [ ]:
```
fig_C1 = go.Figure()

fig_C1.add_trace(go.Box(y=np.array(dfInt100      ['Throughput_C']), name="100"))
fig_C1.add_trace(go.Box(y=np.array(dfInt1000     ['Throughput_C']), name="1 000"))
fig_C1.add_trace(go.Box(y=np.array(dfInt10000    ['Throughput_C']), name="10 000"))
fig_C1.add_trace(go.Box(y=np.array(dfInt100000   ['Throughput_C']), name="100 000")
fig_C1.add_trace(go.Box(y=np.array(dfInt1000000  ['Throughput_C']), name="1M"))
fig_C1.add_trace(go.Box(y=np.array(dfInt2000000  ['Throughput_C']), name="2M"))
fig_C1.add_trace(go.Box(y=np.array(dfInt5000000  ['Throughput_C']), name="5M"))
fig_C1.add_trace(go.Box(y=np.array(dfInt10000000  ['Throughput_C']), name="10M"))
fig_C1.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node C Throughput (Mbps)")

fig_C1.show()
```

In [ ]:
```
fig_C2 = go.Figure()
fig_C2.add_trace(go.Box(y=np.array(dfInt10000000 ['Throughput_C']), name="10M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt20000000 ['Throughput_C']), name="20M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt30000000 ['Throughput_C']), name="30M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt40000000 ['Throughput_C']), name="40M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt50000000 ['Throughput_C']), name="50M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt60000000 ['Throughput_C']), name="60M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt70000000 ['Throughput_C']), name="70M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt80000000 ['Throughput_C']), name="80M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt90000000 ['Throughput_C']), name="90M"))
fig_C2.add_trace(go.Box(y=np.array(dfInt100000000['Throughput_C']), name="100M"))
fig_C2.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Node C Throughput (Mbps)")

fig_C2.show()
```

In [ ]:

```
mean_ThroughputA=[mean_Throughput_Int10000000A,
                  mean_Throughput_Int20000000A,
                  mean_Throughput_Int30000000A,
                  mean_Throughput_Int40000000A,
                  mean_Throughput_Int50000000A,
                  mean_Throughput_Int60000000A,
                  mean_Throughput_Int70000000A,
                  mean_Throughput_Int80000000A,
                  mean_Throughput_Int90000000A,
                  mean_Throughput_Int100000000A]

mean_ThroughputB=[mean_Throughput_Int10000000B,
                  mean_Throughput_Int20000000B,
                  mean_Throughput_Int30000000B,
                  mean_Throughput_Int40000000B,
                  mean_Throughput_Int50000000B,
                  mean_Throughput_Int60000000B,
                  mean_Throughput_Int70000000B,
                  mean_Throughput_Int80000000B,
                  mean_Throughput_Int90000000B,
                  mean_Throughput_Int100000000B]

mean_ThroughputC=[mean_Throughput_Int10000000C,
                  mean_Throughput_Int20000000C,
                  mean_Throughput_Int30000000C,
                  mean_Throughput_Int40000000C,
                  mean_Throughput_Int50000000C,
                  mean_Throughput_Int60000000C,
                  mean_Throughput_Int70000000C,
                  mean_Throughput_Int80000000C,
                  mean_Throughput_Int90000000C,
                  mean_Throughput_Int100000000C]

mean_ThroughputApt2=[mean_Throughput_Int100A,
                     mean_Throughput_Int1000A,
                     mean_Throughput_Int10000A,
                     mean_Throughput_Int100000A,
                     mean_Throughput_Int1000000A,
                     mean_Throughput_Int2000000A,
                     mean_Throughput_Int5000000A,
                     mean_Throughput_Int10000000A]

mean_ThroughputBpt2=[mean_Throughput_Int100B,
                     mean_Throughput_Int1000B,
                     mean_Throughput_Int10000B,
                     mean_Throughput_Int100000B,
                     mean_Throughput_Int1000000B,
                     mean_Throughput_Int2000000B,
                     mean_Throughput_Int5000000B,
                     mean_Throughput_Int10000000B]

mean_ThroughputCpt2=[mean_Throughput_Int100C,
                     mean_Throughput_Int1000C,
                     mean_Throughput_Int10000C,
                     mean_Throughput_Int100000C,
                     mean_Throughput_Int1000000C,
                     mean_Throughput_Int2000000C,
                     mean_Throughput_Int5000000C,
                     mean_Throughput_Int10000000C]
```

```
                    mean_Throughput_int10000000C]
```

In [ ]:
```
x = [10,
     20,
     30,
     40,
     50,
     60,
     70,
     80,
     90,
     100]
y = mean_ThroughputA

fig = plt.figure()
ax = fig.add_subplot(111)

plt.plot(x, y)
plt.plot(x, y, 'bo') # Plotting data
plt.ylim(360,410)
plt.xlim(5,110)

for i, v in enumerate(y):
    ax.text(x[i], v+1.5, "%.2f" %v, ha="left")
plt.xlabel("Interfering Data Rate (Mbps)")
plt.ylabel("Node A Throughput (Mbps)")
plt.show()
```

In [ ]:
```
x2=["100","1 000","10 000","100 000","1M","2M","5M","10M",]
y2=mean_ThroughputApt2

fig = plt.figure()
ax = fig.add_subplot(111)

plt.plot(x2, y2)
plt.plot(range(len(x2)), y2, 'bo') # Plotting data
plt.xticks(range(len(x2)), x2) # Redefining x-axis labels
plt.ylim(404,407.5)
for i, v in enumerate(y2):
    ax.text(i, v+0.15, "%.2f" %v, ha="center")
plt.xlabel("Interfering Data Rate (bps)")
plt.ylabel("Node A Throughput (Mbps)")
plt.show()
```

In [ ]:
```
x3=["10M","20M","30M","40M","50M","60M","70M","80M","90M","100M"]

fig = go.Figure()

fig.add_trace(go.Scatter(x=x3, y=mean_ThroughputA, mode='lines+markers+text', name=
fig.add_trace(go.Scatter(x=x3, y=mean_ThroughputB,mode='lines+markers+text', name='
fig.add_trace(go.Scatter(x=x3, y=mean_ThroughputC,mode='lines+markers+text', name='
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Throughput (Mbps)")

fig.show()
```

In [ ]:

```
x4=["100","1 000","10 000","100 000","1M","2M","5M","10M",]

fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=y2, mode='lines+markers+text', name='Node A', text
fig.add_trace(go.Scatter(x=x4, y=mean_ThroughputBpt2,mode='lines+markers+text', nam
fig.add_trace(go.Scatter(x=x4, y=mean_ThroughputCpt2,mode='lines+markers+text', nam
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Throughput (Mbps)")

fig.show()
```

In [ ]:

```
mean_Rx_PacketsA=[mean_Rx_Packets_Int10000000A,
                  mean_Rx_Packets_Int20000000A,
                  mean_Rx_Packets_Int30000000A,
                  mean_Rx_Packets_Int40000000A,
                  mean_Rx_Packets_Int50000000A,
                  mean_Rx_Packets_Int60000000A,
                  mean_Rx_Packets_Int70000000A,
                  mean_Rx_Packets_Int80000000A,
                  mean_Rx_Packets_Int90000000A,
                  mean_Rx_Packets_Int100000000A]

mean_Rx_PacketsB=[mean_Rx_Packets_Int10000000B,
                  mean_Rx_Packets_Int20000000B,
                  mean_Rx_Packets_Int30000000B,
                  mean_Rx_Packets_Int40000000B,
                  mean_Rx_Packets_Int50000000B,
                  mean_Rx_Packets_Int60000000B,
                  mean_Rx_Packets_Int70000000B,
                  mean_Rx_Packets_Int80000000B,
                  mean_Rx_Packets_Int90000000B,
                  mean_Rx_Packets_Int100000000B]

mean_Rx_PacketsC=[mean_Rx_Packets_Int10000000C,
                  mean_Rx_Packets_Int20000000C,
                  mean_Rx_Packets_Int30000000C,
                  mean_Rx_Packets_Int40000000C,
                  mean_Rx_Packets_Int50000000C,
                  mean_Rx_Packets_Int60000000C,
                  mean_Rx_Packets_Int70000000C,
                  mean_Rx_Packets_Int80000000C,
                  mean_Rx_Packets_Int90000000C,
                  mean_Rx_Packets_Int100000000C]
mean_Rx_PacketsApt2=[mean_Rx_Packets_Int100A,
                     mean_Rx_Packets_Int1000A,
                     mean_Rx_Packets_Int10000A,
                     mean_Rx_Packets_Int100000A,
                     mean_Rx_Packets_Int1000000A,
                     mean_Rx_Packets_Int2000000A,
                     mean_Rx_Packets_Int5000000A,
                     mean_Rx_Packets_Int10000000A]
mean_Rx_PacketsBpt2=[mean_Rx_Packets_Int100B,
                     mean_Rx_Packets_Int1000B,
                     mean_Rx_Packets_Int10000B,
                     mean_Rx_Packets_Int100000B,
                     mean_Rx_Packets_Int1000000B,
                     mean_Rx_Packets_Int2000000B,
                     mean_Rx_Packets_Int5000000B,
                     mean_Rx_Packets_Int10000000B]
mean_Rx_PacketsCpt2=[mean_Rx_Packets_Int100C,
                     mean_Rx_Packets_Int1000C,
                     mean_Rx_Packets_Int10000C,
                     mean_Rx_Packets_Int100000C,
                     mean_Rx_Packets_Int1000000C,
                     mean_Rx_Packets_Int2000000C,
                     mean_Rx_Packets_Int5000000C,
                     mean_Rx_Packets_Int10000000C]
```

In [ ]:

```
mean_Rx_BytesA=[mean_Rx_Bytes_Int10000000A,
                mean_Rx_Bytes_Int20000000A,
                mean_Rx_Bytes_Int30000000A,
                mean_Rx_Bytes_Int40000000A,
                mean_Rx_Bytes_Int50000000A,
                mean_Rx_Bytes_Int60000000A,
                mean_Rx_Bytes_Int70000000A,
                mean_Rx_Bytes_Int80000000A,
                mean_Rx_Bytes_Int90000000A,
                mean_Rx_Bytes_Int100000000A]

mean_Rx_BytesB=[mean_Rx_Bytes_Int10000000B,
                mean_Rx_Bytes_Int20000000B,
                mean_Rx_Bytes_Int30000000B,
                mean_Rx_Bytes_Int40000000B,
                mean_Rx_Bytes_Int50000000B,
                mean_Rx_Bytes_Int60000000B,
                mean_Rx_Bytes_Int70000000B,
                mean_Rx_Bytes_Int80000000B,
                mean_Rx_Bytes_Int90000000B,
                mean_Rx_Bytes_Int100000000B]

mean_Rx_BytesC=[mean_Rx_Bytes_Int10000000C,
                mean_Rx_Bytes_Int20000000C,
                mean_Rx_Bytes_Int30000000C,
                mean_Rx_Bytes_Int40000000C,
                mean_Rx_Bytes_Int50000000C,
                mean_Rx_Bytes_Int60000000C,
                mean_Rx_Bytes_Int70000000C,
                mean_Rx_Bytes_Int80000000C,
                mean_Rx_Bytes_Int90000000C,
                mean_Rx_Bytes_Int100000000C]
mean_Rx_BytesApt2=[mean_Rx_Bytes_Int100A,
                   mean_Rx_Bytes_Int1000A,
                   mean_Rx_Bytes_Int10000A,
                   mean_Rx_Bytes_Int100000A,
                   mean_Rx_Bytes_Int1000000A,
                   mean_Rx_Bytes_Int2000000A,
                   mean_Rx_Bytes_Int5000000A,
                   mean_Rx_Bytes_Int10000000A]
mean_Rx_BytesBpt2=[mean_Rx_Bytes_Int100B,
                   mean_Rx_Bytes_Int1000B,
                   mean_Rx_Bytes_Int10000B,
                   mean_Rx_Bytes_Int100000B,
                   mean_Rx_Bytes_Int1000000B,
                   mean_Rx_Bytes_Int2000000B,
                   mean_Rx_Bytes_Int5000000B,
                   mean_Rx_Bytes_Int10000000B]
mean_Rx_BytesCpt2=[mean_Rx_Bytes_Int100C,
                   mean_Rx_Bytes_Int1000C,
                   mean_Rx_Bytes_Int10000C,
                   mean_Rx_Bytes_Int100000C,
                   mean_Rx_Bytes_Int1000000C,
                   mean_Rx_Bytes_Int2000000C,
                   mean_Rx_Bytes_Int5000000C,
                   mean_Rx_Bytes_Int10000000C]
```

In [ ]:

```
Tx_BytesA=[Tx_Bytes_Int10000000A,
                   Tx_Bytes_Int20000000A,
                   Tx_Bytes_Int30000000A,
                   Tx_Bytes_Int40000000A,
                   Tx_Bytes_Int50000000A,
                   Tx_Bytes_Int60000000A,
                   Tx_Bytes_Int70000000A,
                   Tx_Bytes_Int80000000A,
                   Tx_Bytes_Int90000000A,
                   Tx_Bytes_Int100000000A]

Tx_BytesB=[Tx_Bytes_Int10000000B,
                   Tx_Bytes_Int20000000B,
                   Tx_Bytes_Int30000000B,
                   Tx_Bytes_Int40000000B,
                   Tx_Bytes_Int50000000B,
                   Tx_Bytes_Int60000000B,
                   Tx_Bytes_Int70000000B,
                   Tx_Bytes_Int80000000B,
                   Tx_Bytes_Int90000000B,
                   Tx_Bytes_Int100000000B]

Tx_BytesC=[Tx_Bytes_Int10000000C,
                   Tx_Bytes_Int20000000C,
                   Tx_Bytes_Int30000000C,
                   Tx_Bytes_Int40000000C,
                   Tx_Bytes_Int50000000C,
                   Tx_Bytes_Int60000000C,
                   Tx_Bytes_Int70000000C,
                   Tx_Bytes_Int80000000C,
                   Tx_Bytes_Int90000000C,
                   Tx_Bytes_Int100000000C]
Tx_BytesApt2=[Tx_Bytes_Int100A,
                    Tx_Bytes_Int1000A,
                    Tx_Bytes_Int10000A,
                    Tx_Bytes_Int100000A,
                    Tx_Bytes_Int1000000A,
                    Tx_Bytes_Int2000000A,
                    Tx_Bytes_Int5000000A,
                    Tx_Bytes_Int10000000A]
Tx_BytesBpt2=[Tx_Bytes_Int100B,
                    Tx_Bytes_Int1000B,
                    Tx_Bytes_Int10000B,
                    Tx_Bytes_Int100000B,
                    Tx_Bytes_Int1000000B,
                    Tx_Bytes_Int2000000B,
                    Tx_Bytes_Int5000000B,
                    Tx_Bytes_Int10000000B]
Tx_BytesCpt2=[Tx_Bytes_Int100C,
                    Tx_Bytes_Int1000C,
                    Tx_Bytes_Int10000C,
                    Tx_Bytes_Int100000C,
                    Tx_Bytes_Int1000000C,
                    Tx_Bytes_Int2000000C,
                    Tx_Bytes_Int5000000C,
                    Tx_Bytes_Int10000000C]
```

In [ ]:

```
Tx_PacketsA=[Tx_Packets_Int10000000A,
                  Tx_Packets_Int20000000A,
                  Tx_Packets_Int30000000A,
                  Tx_Packets_Int40000000A,
                  Tx_Packets_Int50000000A,
                  Tx_Packets_Int60000000A,
                  Tx_Packets_Int70000000A,
                  Tx_Packets_Int80000000A,
                  Tx_Packets_Int90000000A,
                  Tx_Packets_Int100000000A]

Tx_PacketsB=[Tx_Packets_Int10000000B,
                  Tx_Packets_Int20000000B,
                  Tx_Packets_Int30000000B,
                  Tx_Packets_Int40000000B,
                  Tx_Packets_Int50000000B,
                  Tx_Packets_Int60000000B,
                  Tx_Packets_Int70000000B,
                  Tx_Packets_Int80000000B,
                  Tx_Packets_Int90000000B,
                  Tx_Packets_Int100000000B]

Tx_PacketsC=[Tx_Packets_Int10000000C,
                  Tx_Packets_Int20000000C,
                  Tx_Packets_Int30000000C,
                  Tx_Packets_Int40000000C,
                  Tx_Packets_Int50000000C,
                  Tx_Packets_Int60000000C,
                  Tx_Packets_Int70000000C,
                  Tx_Packets_Int80000000C,
                  Tx_Packets_Int90000000C,
                  Tx_Packets_Int100000000C]
Tx_PacketsApt2=[Tx_Packets_Int100A,
                   Tx_Packets_Int1000A,
                   Tx_Packets_Int10000A,
                   Tx_Packets_Int100000A,
                   Tx_Packets_Int1000000A,
                   Tx_Packets_Int2000000A,
                   Tx_Packets_Int5000000A,
                   Tx_Packets_Int10000000A]
Tx_PacketsBpt2=[Tx_Packets_Int100B,
                   Tx_Packets_Int1000B,
                   Tx_Packets_Int10000B,
                   Tx_Packets_Int100000B,
                   Tx_Packets_Int1000000B,
                   Tx_Packets_Int2000000B,
                   Tx_Packets_Int5000000B,
                   Tx_Packets_Int10000000B]
Tx_PacketsCpt2=[Tx_Packets_Int100C,
                   Tx_Packets_Int1000C,
                   Tx_Packets_Int10000C,
                   Tx_Packets_Int100000C,
                   Tx_Packets_Int1000000C,
                   Tx_Packets_Int2000000C,
                   Tx_Packets_Int5000000C,
                   Tx_Packets_Int10000000C]
```

In [ ]:

```
Tx_OfferedA=[Tx_Offered_Int10000000A,
                    Tx_Offered_Int20000000A,
                    Tx_Offered_Int30000000A,
                    Tx_Offered_Int40000000A,
                    Tx_Offered_Int50000000A,
                    Tx_Offered_Int60000000A,
                    Tx_Offered_Int70000000A,
                    Tx_Offered_Int80000000A,
                    Tx_Offered_Int90000000A,
                    Tx_Offered_Int100000000A]

Tx_OfferedB=[Tx_Offered_Int10000000B,
                    Tx_Offered_Int20000000B,
                    Tx_Offered_Int30000000B,
                    Tx_Offered_Int40000000B,
                    Tx_Offered_Int50000000B,
                    Tx_Offered_Int60000000B,
                    Tx_Offered_Int70000000B,
                    Tx_Offered_Int80000000B,
                    Tx_Offered_Int90000000B,
                    Tx_Offered_Int100000000B]

Tx_OfferedC=[Tx_Offered_Int10000000C,
                    Tx_Offered_Int20000000C,
                    Tx_Offered_Int30000000C,
                    Tx_Offered_Int40000000C,
                    Tx_Offered_Int50000000C,
                    Tx_Offered_Int60000000C,
                    Tx_Offered_Int70000000C,
                    Tx_Offered_Int80000000C,
                    Tx_Offered_Int90000000C,
                    Tx_Offered_Int100000000C]
Tx_OfferedApt2=[Tx_Offered_Int100A,
                     Tx_Offered_Int1000A,
                     Tx_Offered_Int10000A,
                     Tx_Offered_Int100000A,
                     Tx_Offered_Int1000000A,
                     Tx_Offered_Int2000000A,
                     Tx_Offered_Int5000000A,
                     Tx_Offered_Int10000000A]
Tx_OfferedBpt2=[Tx_Offered_Int100B,
                     Tx_Offered_Int1000B,
                     Tx_Offered_Int10000B,
                     Tx_Offered_Int100000B,
                     Tx_Offered_Int1000000B,
                     Tx_Offered_Int2000000B,
                     Tx_Offered_Int5000000B,
                     Tx_Offered_Int10000000B]
Tx_OfferedCpt2=[Tx_Offered_Int100C,
                     Tx_Offered_Int1000C,
                     Tx_Offered_Int10000C,
                     Tx_Offered_Int100000C,
                     Tx_Offered_Int1000000C,
                     Tx_Offered_Int2000000C,
                     Tx_Offered_Int5000000C,
                     Tx_Offered_Int10000000C]
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=Tx_OfferedApt2, mode='lines+markers+text', name='N
fig.add_trace(go.Scatter(x=x4, y=Tx_OfferedBpt2,mode='lines+markers+text', name='No
fig.add_trace(go.Scatter(x=x4, y=Tx_OfferedCpt2,mode='lines+markers+text', name='No
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Tx Offered (Mbps)")
fig.show()

fig = go.Figure()

fig.add_trace(go.Scatter(x=x3, y=Tx_OfferedA, mode='lines+markers+text', name='Node
fig.add_trace(go.Scatter(x=x3, y=Tx_OfferedB,mode='lines+markers+text', name='Node
fig.add_trace(go.Scatter(x=x3, y=Tx_OfferedC,mode='lines+markers+text', name='Node
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Tx Offered (Mbps)")

fig.show()
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x3, y=Tx_BytesA, mode='lines+markers+text', name='Node A
fig.add_trace(go.Scatter(x=x3, y=Tx_BytesB,mode='lines+markers+text', name='Node B
fig.add_trace(go.Scatter(x=x3, y=Tx_BytesC,mode='lines+markers+text', name='Node C
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Transmitted Bytes")

fig.show()

fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=Tx_BytesApt2, mode='lines+markers+text', name='Nod
fig.add_trace(go.Scatter(x=x4, y=Tx_BytesBpt2,mode='lines+markers+text', name='Node
fig.add_trace(go.Scatter(x=x4, y=Tx_BytesCpt2,mode='lines+markers+text', name='Node
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Transmitted Bytes")

fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x3, y=Tx_PacketsA, mode='lines+markers+text', name='Node
fig.add_trace(go.Scatter(x=x3, y=Tx_PacketsB,mode='lines+markers+text', name='Node
fig.add_trace(go.Scatter(x=x3, y=Tx_PacketsC,mode='lines+markers+text', name='Node
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Transmitted Packets")

fig.show()

fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=Tx_PacketsApt2, mode='lines+markers+text', name='N
fig.add_trace(go.Scatter(x=x4, y=Tx_PacketsBpt2,mode='lines+markers+text', name='No
fig.add_trace(go.Scatter(x=x4, y=Tx_PacketsCpt2,mode='lines+markers+text', name='No
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Transmitted Packets")

fig.show()
```

In [ ]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=x3, y=mean_Rx_PacketsA, mode='lines+markers+text', name=
fig.add_trace(go.Scatter(x=x3, y=mean_Rx_PacketsB,mode='lines+markers+text', name='
fig.add_trace(go.Scatter(x=x3, y=mean_Rx_PacketsC,mode='lines+markers+text', name='
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Average # Packets Received")

fig.show()

fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=mean_Rx_PacketsApt2, mode='lines+markers+text', na
fig.add_trace(go.Scatter(x=x4, y=mean_Rx_PacketsBpt2,mode='lines+markers+text', nam
fig.add_trace(go.Scatter(x=x4, y=mean_Rx_PacketsCpt2,mode='lines+markers+text', nam
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Average # Packets Received")

fig.show()
```

In [ ]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=x3, y=mean_Rx_BytesA, mode='lines+markers+text', name='N
fig.add_trace(go.Scatter(x=x3, y=mean_Rx_BytesB,mode='lines+markers+text', name='No
fig.add_trace(go.Scatter(x=x3, y=mean_Rx_BytesC,mode='lines+markers+text', name='No
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Average # Received Bytes")

fig.show()

fig = go.Figure()

fig.add_trace(go.Scatter(x=x4, y=mean_Rx_BytesApt2, mode='lines+markers+text', name
fig.add_trace(go.Scatter(x=x4, y=mean_Rx_BytesBpt2,mode='lines+markers+text', name=
fig.add_trace(go.Scatter(x=x4, y=mean_Rx_BytesCpt2,mode='lines+markers+text', name=
fig.update_layout(xaxis_title="Interfering Data Rate (bps)",
                  yaxis_title="Average # Received Bytes")

fig.show()
```

Interfering Distance.ipynb

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import statistics
import scipy.stats
import plotly.graph_objects as go
import plotly.express as px
from scipy.interpolate import interp1d
```

In [ ]:
```
dfd2 = pd.read_csv('Interfering_10000000.csv')


mean_Rx_Packets_d2A = sum(dfd2['Rx_Packets_A']) / len(dfd2['Rx_Packets_A'])
mean_Rx_Bytes_d2A = sum(dfd2['Rx_Bytes_A']) / len(dfd2['Rx_Bytes_A'])
mean_Throughput_d2A = sum(dfd2['Throughput_A']) / len(dfd2['Throughput_A'])

Tx_Packets_d2A = dfd2['Tx_Packets_A'][0]
Tx_Bytes_d2A = dfd2['Tx_Bytes_A'][0]
Tx_Offered_d2A= dfd2['TxOffered_A'][0]


mean_Rx_Packets_d2B = sum(dfd2['Rx_Packets_B']) / len(dfd2['Rx_Packets_B'])
mean_Rx_Bytes_d2B = sum(dfd2['Rx_Bytes_B']) / len(dfd2['Rx_Bytes_B'])
mean_Throughput_d2B = sum(dfd2['Throughput_B']) / len(dfd2['Throughput_B'])

mean_Rx_Packets_d2C = sum(dfd2['Rx_Packets_C']) / len(dfd2['Rx_Packets_C'])
mean_Rx_Bytes_d2C = sum(dfd2['Rx_Bytes_C']) / len(dfd2['Rx_Bytes_C'])
mean_Throughput_d2C = sum(dfd2['Throughput_C']) / len(dfd2['Throughput_C'])



Tx_Packets_d2B = dfd2['Tx_Packets_B'][0]
Tx_Bytes_d2B = dfd2['Tx_Bytes_B'][0]
Tx_Offered_d2B= dfd2['TxOffered_B'][0]

Tx_Packets_d2C = dfd2['Tx_Packets_C'][0]
Tx_Bytes_d2C = dfd2['Tx_Bytes_C'][0]
Tx_Offered_d2C= dfd2['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d2A,'Tx_Bytes_A:',Tx_Bytes_d2A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d2A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d2B,'Tx_Bytes_B:',Tx_Bytes_d2B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d2B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d2C,'Tx_Bytes_C:',Tx_Bytes_d2C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d2C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd2
```

In [ ]:

```
dfd3 = pd.read_csv('Interfering_d3.csv')


mean_Rx_Packets_d3A = sum(dfd3['Rx_Packets_A']) / len(dfd3['Rx_Packets_A'])
mean_Rx_Bytes_d3A = sum(dfd3['Rx_Bytes_A']) / len(dfd3['Rx_Bytes_A'])
mean_Throughput_d3A = sum(dfd3['Throughput_A']) / len(dfd3['Throughput_A'])

Tx_Packets_d3A = dfd3['Tx_Packets_A'][0]
Tx_Bytes_d3A = dfd3['Tx_Bytes_A'][0]
Tx_Offered_d3A= dfd3['TxOffered_A'][0]


mean_Rx_Packets_d3B = sum(dfd3['Rx_Packets_B']) / len(dfd3['Rx_Packets_B'])
mean_Rx_Bytes_d3B = sum(dfd3['Rx_Bytes_B']) / len(dfd3['Rx_Bytes_B'])
mean_Throughput_d3B = sum(dfd3['Throughput_B']) / len(dfd3['Throughput_B'])

mean_Rx_Packets_d3C = sum(dfd3['Rx_Packets_C']) / len(dfd3['Rx_Packets_C'])
mean_Rx_Bytes_d3C = sum(dfd3['Rx_Bytes_C']) / len(dfd3['Rx_Bytes_C'])
mean_Throughput_d3C = sum(dfd3['Throughput_C']) / len(dfd3['Throughput_C'])



Tx_Packets_d3B = dfd3['Tx_Packets_B'][0]
Tx_Bytes_d3B = dfd3['Tx_Bytes_B'][0]
Tx_Offered_d3B= dfd3['TxOffered_B'][0]

Tx_Packets_d3C = dfd3['Tx_Packets_C'][0]
Tx_Bytes_d3C = dfd3['Tx_Bytes_C'][0]
Tx_Offered_d3C= dfd3['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d3A,'Tx_Bytes_A:',Tx_Bytes_d3A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d3A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d3B,'Tx_Bytes_B:',Tx_Bytes_d3B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d3B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d3C,'Tx_Bytes_C:',Tx_Bytes_d3C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d3C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd3
```

In [ ]:

```
dfd4 = pd.read_csv('Interfering_d4.csv')


mean_Rx_Packets_d4A = sum(dfd4['Rx_Packets_A']) / len(dfd4['Rx_Packets_A'])
mean_Rx_Bytes_d4A = sum(dfd4['Rx_Bytes_A']) / len(dfd4['Rx_Bytes_A'])
mean_Throughput_d4A = sum(dfd4['Throughput_A']) / len(dfd4['Throughput_A'])

Tx_Packets_d4A = dfd4['Tx_Packets_A'][0]
Tx_Bytes_d4A = dfd4['Tx_Bytes_A'][0]
Tx_Offered_d4A= dfd4['TxOffered_A'][0]


mean_Rx_Packets_d4B = sum(dfd4['Rx_Packets_B']) / len(dfd4['Rx_Packets_B'])
mean_Rx_Bytes_d4B = sum(dfd4['Rx_Bytes_B']) / len(dfd4['Rx_Bytes_B'])
mean_Throughput_d4B = sum(dfd4['Throughput_B']) / len(dfd4['Throughput_B'])

mean_Rx_Packets_d4C = sum(dfd4['Rx_Packets_C']) / len(dfd4['Rx_Packets_C'])
mean_Rx_Bytes_d4C = sum(dfd4['Rx_Bytes_C']) / len(dfd4['Rx_Bytes_C'])
mean_Throughput_d4C = sum(dfd4['Throughput_C']) / len(dfd4['Throughput_C'])



Tx_Packets_d4B = dfd4['Tx_Packets_B'][0]
Tx_Bytes_d4B = dfd4['Tx_Bytes_B'][0]
Tx_Offered_d4B= dfd4['TxOffered_B'][0]

Tx_Packets_d4C = dfd4['Tx_Packets_C'][0]
Tx_Bytes_d4C = dfd4['Tx_Bytes_C'][0]
Tx_Offered_d4C= dfd4['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d4A,'Tx_Bytes_A:',Tx_Bytes_d4A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d4A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d4B,'Tx_Bytes_B:',Tx_Bytes_d4B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d4B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d4C,'Tx_Bytes_C:',Tx_Bytes_d4C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d4C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd4
```

In [ ]:

```
dfd5 = pd.read_csv('Interfering_d5.csv')


mean_Rx_Packets_d5A = sum(dfd5['Rx_Packets_A']) / len(dfd5['Rx_Packets_A'])
mean_Rx_Bytes_d5A = sum(dfd5['Rx_Bytes_A']) / len(dfd5['Rx_Bytes_A'])
mean_Throughput_d5A = sum(dfd5['Throughput_A']) / len(dfd5['Throughput_A'])

Tx_Packets_d5A = dfd5['Tx_Packets_A'][0]
Tx_Bytes_d5A = dfd5['Tx_Bytes_A'][0]
Tx_Offered_d5A= dfd5['TxOffered_A'][0]


mean_Rx_Packets_d5B = sum(dfd5['Rx_Packets_B']) / len(dfd5['Rx_Packets_B'])
mean_Rx_Bytes_d5B = sum(dfd5['Rx_Bytes_B']) / len(dfd5['Rx_Bytes_B'])
mean_Throughput_d5B = sum(dfd5['Throughput_B']) / len(dfd5['Throughput_B'])

mean_Rx_Packets_d5C = sum(dfd5['Rx_Packets_C']) / len(dfd5['Rx_Packets_C'])
mean_Rx_Bytes_d5C = sum(dfd5['Rx_Bytes_C']) / len(dfd5['Rx_Bytes_C'])
mean_Throughput_d5C = sum(dfd5['Throughput_C']) / len(dfd5['Throughput_C'])



Tx_Packets_d5B = dfd5['Tx_Packets_B'][0]
Tx_Bytes_d5B = dfd5['Tx_Bytes_B'][0]
Tx_Offered_d5B= dfd5['TxOffered_B'][0]

Tx_Packets_d5C = dfd5['Tx_Packets_C'][0]
Tx_Bytes_d5C = dfd5['Tx_Bytes_C'][0]
Tx_Offered_d5C= dfd5['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d5A,'Tx_Bytes_A:',Tx_Bytes_d5A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d5A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d5B,'Tx_Bytes_B:',Tx_Bytes_d5B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d5B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d5C,'Tx_Bytes_C:',Tx_Bytes_d5C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d5C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd5
```

In [ ]:

```
dfd6 = pd.read_csv('Interfering_d6.csv')


mean_Rx_Packets_d6A = sum(dfd6['Rx_Packets_A']) / len(dfd6['Rx_Packets_A'])
mean_Rx_Bytes_d6A = sum(dfd6['Rx_Bytes_A']) / len(dfd6['Rx_Bytes_A'])
mean_Throughput_d6A = sum(dfd6['Throughput_A']) / len(dfd6['Throughput_A'])

Tx_Packets_d6A = dfd6['Tx_Packets_A'][0]
Tx_Bytes_d6A = dfd6['Tx_Bytes_A'][0]
Tx_Offered_d6A= dfd6['TxOffered_A'][0]


mean_Rx_Packets_d6B = sum(dfd6['Rx_Packets_B']) / len(dfd6['Rx_Packets_B'])
mean_Rx_Bytes_d6B = sum(dfd6['Rx_Bytes_B']) / len(dfd6['Rx_Bytes_B'])
mean_Throughput_d6B = sum(dfd6['Throughput_B']) / len(dfd6['Throughput_B'])

mean_Rx_Packets_d6C = sum(dfd6['Rx_Packets_C']) / len(dfd6['Rx_Packets_C'])
mean_Rx_Bytes_d6C = sum(dfd6['Rx_Bytes_C']) / len(dfd6['Rx_Bytes_C'])
mean_Throughput_d6C = sum(dfd6['Throughput_C']) / len(dfd6['Throughput_C'])



Tx_Packets_d6B = dfd6['Tx_Packets_B'][0]
Tx_Bytes_d6B = dfd6['Tx_Bytes_B'][0]
Tx_Offered_d6B= dfd6['TxOffered_B'][0]

Tx_Packets_d6C = dfd6['Tx_Packets_C'][0]
Tx_Bytes_d6C = dfd6['Tx_Bytes_C'][0]
Tx_Offered_d6C= dfd6['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d6A,'Tx_Bytes_A:',Tx_Bytes_d6A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d6A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d6B,'Tx_Bytes_B:',Tx_Bytes_d6B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d6B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d6C,'Tx_Bytes_C:',Tx_Bytes_d6C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d6C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd6
```

In [ ]:

```
dfd7 = pd.read_csv('Interfering_d7.csv')


mean_Rx_Packets_d7A = sum(dfd7['Rx_Packets_A']) / len(dfd7['Rx_Packets_A'])
mean_Rx_Bytes_d7A = sum(dfd7['Rx_Bytes_A']) / len(dfd7['Rx_Bytes_A'])
mean_Throughput_d7A = sum(dfd7['Throughput_A']) / len(dfd7['Throughput_A'])

Tx_Packets_d7A = dfd7['Tx_Packets_A'][0]
Tx_Bytes_d7A = dfd7['Tx_Bytes_A'][0]
Tx_Offered_d7A= dfd7['TxOffered_A'][0]


mean_Rx_Packets_d7B = sum(dfd7['Rx_Packets_B']) / len(dfd7['Rx_Packets_B'])
mean_Rx_Bytes_d7B = sum(dfd7['Rx_Bytes_B']) / len(dfd7['Rx_Bytes_B'])
mean_Throughput_d7B = sum(dfd7['Throughput_B']) / len(dfd7['Throughput_B'])

mean_Rx_Packets_d7C = sum(dfd7['Rx_Packets_C']) / len(dfd7['Rx_Packets_C'])
mean_Rx_Bytes_d7C = sum(dfd7['Rx_Bytes_C']) / len(dfd7['Rx_Bytes_C'])
mean_Throughput_d7C = sum(dfd7['Throughput_C']) / len(dfd7['Throughput_C'])



Tx_Packets_d7B = dfd7['Tx_Packets_B'][0]
Tx_Bytes_d7B = dfd7['Tx_Bytes_B'][0]
Tx_Offered_d7B= dfd7['TxOffered_B'][0]

Tx_Packets_d7C = dfd7['Tx_Packets_C'][0]
Tx_Bytes_d7C = dfd7['Tx_Bytes_C'][0]
Tx_Offered_d7C= dfd7['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d7A,'Tx_Bytes_A:',Tx_Bytes_d7A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d7A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d7B,'Tx_Bytes_B:',Tx_Bytes_d7B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d7B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d7C,'Tx_Bytes_C:',Tx_Bytes_d7C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d7C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd7
```

In [ ]:

```
dfd8 = pd.read_csv('Interfering_d8.csv')


mean_Rx_Packets_d8A = sum(dfd8['Rx_Packets_A']) / len(dfd8['Rx_Packets_A'])
mean_Rx_Bytes_d8A = sum(dfd8['Rx_Bytes_A']) / len(dfd8['Rx_Bytes_A'])
mean_Throughput_d8A = sum(dfd8['Throughput_A']) / len(dfd8['Throughput_A'])

Tx_Packets_d8A = dfd8['Tx_Packets_A'][0]
Tx_Bytes_d8A = dfd8['Tx_Bytes_A'][0]
Tx_Offered_d8A= dfd8['TxOffered_A'][0]


mean_Rx_Packets_d8B = sum(dfd8['Rx_Packets_B']) / len(dfd8['Rx_Packets_B'])
mean_Rx_Bytes_d8B = sum(dfd8['Rx_Bytes_B']) / len(dfd8['Rx_Bytes_B'])
mean_Throughput_d8B = sum(dfd8['Throughput_B']) / len(dfd8['Throughput_B'])

mean_Rx_Packets_d8C = sum(dfd8['Rx_Packets_C']) / len(dfd8['Rx_Packets_C'])
mean_Rx_Bytes_d8C = sum(dfd8['Rx_Bytes_C']) / len(dfd8['Rx_Bytes_C'])
mean_Throughput_d8C = sum(dfd8['Throughput_C']) / len(dfd8['Throughput_C'])



Tx_Packets_d8B = dfd8['Tx_Packets_B'][0]
Tx_Bytes_d8B = dfd8['Tx_Bytes_B'][0]
Tx_Offered_d8B= dfd8['TxOffered_B'][0]

Tx_Packets_d8C = dfd8['Tx_Packets_C'][0]
Tx_Bytes_d8C = dfd8['Tx_Bytes_C'][0]
Tx_Offered_d8C= dfd8['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d8A,'Tx_Bytes_A:',Tx_Bytes_d8A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d8A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d8B,'Tx_Bytes_B:',Tx_Bytes_d8B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d8B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d8C,'Tx_Bytes_C:',Tx_Bytes_d8C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d8C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd8
```

In [ ]:

```
dfd9 = pd.read_csv('Interfering_d9.csv')


mean_Rx_Packets_d9A = sum(dfd9['Rx_Packets_A']) / len(dfd9['Rx_Packets_A'])
mean_Rx_Bytes_d9A = sum(dfd9['Rx_Bytes_A']) / len(dfd9['Rx_Bytes_A'])
mean_Throughput_d9A = sum(dfd9['Throughput_A']) / len(dfd9['Throughput_A'])

Tx_Packets_d9A = dfd9['Tx_Packets_A'][0]
Tx_Bytes_d9A = dfd9['Tx_Bytes_A'][0]
Tx_Offered_d9A= dfd9['TxOffered_A'][0]


mean_Rx_Packets_d9B = sum(dfd9['Rx_Packets_B']) / len(dfd9['Rx_Packets_B'])
mean_Rx_Bytes_d9B = sum(dfd9['Rx_Bytes_B']) / len(dfd9['Rx_Bytes_B'])
mean_Throughput_d9B = sum(dfd9['Throughput_B']) / len(dfd9['Throughput_B'])

mean_Rx_Packets_d9C = sum(dfd9['Rx_Packets_C']) / len(dfd9['Rx_Packets_C'])
mean_Rx_Bytes_d9C = sum(dfd9['Rx_Bytes_C']) / len(dfd9['Rx_Bytes_C'])
mean_Throughput_d9C = sum(dfd9['Throughput_C']) / len(dfd9['Throughput_C'])



Tx_Packets_d9B = dfd9['Tx_Packets_B'][0]
Tx_Bytes_d9B = dfd9['Tx_Bytes_B'][0]
Tx_Offered_d9B= dfd9['TxOffered_B'][0]

Tx_Packets_d9C = dfd9['Tx_Packets_C'][0]
Tx_Bytes_d9C = dfd9['Tx_Bytes_C'][0]
Tx_Offered_d9C= dfd9['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d9A,'Tx_Bytes_A:',Tx_Bytes_d9A, ' Tx_Offered_A:',
       'mean_Rx_Packets_A:',mean_Rx_Packets_d9A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_d

print ('Tx_Packets_B:',Tx_Packets_d9B,'Tx_Bytes_B:',Tx_Bytes_d9B, ' Tx_Offered_B:',
       'mean_Rx_Packets_B:',mean_Rx_Packets_d9B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_d

print ('Tx_Packets_C:',Tx_Packets_d9C,'Tx_Bytes_C:',Tx_Bytes_d9C, ' Tx_Offered_C:',
       'mean_Rx_Packets_C:',mean_Rx_Packets_d9C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_d

dfd9
```

In [ ]:
```
dfd10 = pd.read_csv('Interfering_d10.csv')


mean_Rx_Packets_d10A = sum(dfd10['Rx_Packets_A']) / len(dfd10['Rx_Packets_A'])
mean_Rx_Bytes_d10A = sum(dfd10['Rx_Bytes_A']) / len(dfd10['Rx_Bytes_A'])
mean_Throughput_d10A = sum(dfd10['Throughput_A']) / len(dfd10['Throughput_A'])

Tx_Packets_d10A = dfd10['Tx_Packets_A'][0]
Tx_Bytes_d10A = dfd10['Tx_Bytes_A'][0]
Tx_Offered_d10A= dfd10['TxOffered_A'][0]


mean_Rx_Packets_d10B = sum(dfd10['Rx_Packets_B']) / len(dfd10['Rx_Packets_B'])
mean_Rx_Bytes_d10B = sum(dfd10['Rx_Bytes_B']) / len(dfd10['Rx_Bytes_B'])
mean_Throughput_d10B = sum(dfd10['Throughput_B']) / len(dfd10['Throughput_B'])

mean_Rx_Packets_d10C = sum(dfd10['Rx_Packets_C']) / len(dfd10['Rx_Packets_C'])
mean_Rx_Bytes_d10C = sum(dfd10['Rx_Bytes_C']) / len(dfd10['Rx_Bytes_C'])
mean_Throughput_d10C = sum(dfd10['Throughput_C']) / len(dfd10['Throughput_C'])



Tx_Packets_d10B = dfd10['Tx_Packets_B'][0]
Tx_Bytes_d10B = dfd10['Tx_Bytes_B'][0]
Tx_Offered_d10B= dfd10['TxOffered_B'][0]

Tx_Packets_d10C = dfd10['Tx_Packets_C'][0]
Tx_Bytes_d10C = dfd10['Tx_Bytes_C'][0]
Tx_Offered_d10C= dfd10['TxOffered_C'][0]

print ('Tx_Packets_A:',Tx_Packets_d10A,'Tx_Bytes_A:',Tx_Bytes_d10A, ' Tx_Offered_A:
       'mean_Rx_Packets_A:',mean_Rx_Packets_d10A,' mean_Rx_Bytes_A:',mean_Rx_Bytes_

print ('Tx_Packets_B:',Tx_Packets_d10B,'Tx_Bytes_B:',Tx_Bytes_d10B, ' Tx_Offered_B:
       'mean_Rx_Packets_B:',mean_Rx_Packets_d10B,' mean_Rx_Bytes_B:',mean_Rx_Bytes_

print ('Tx_Packets_C:',Tx_Packets_d10C,'Tx_Bytes_C:',Tx_Bytes_d10C, ' Tx_Offered_C:
       'mean_Rx_Packets_C:',mean_Rx_Packets_d10C,' mean_Rx_Bytes_C:',mean_Rx_Bytes_

dfd10
```

In [ ]:
```
fig_A1 = go.Figure()

fig_A1.add_trace(go.Box(y=np.array(dfd2   ['Throughput_A']), name=2))
fig_A1.add_trace(go.Box(y=np.array(dfd3   ['Throughput_A']), name=3))
fig_A1.add_trace(go.Box(y=np.array(dfd4   ['Throughput_A']), name=4))
fig_A1.add_trace(go.Box(y=np.array(dfd5   ['Throughput_A']), name=5))
fig_A1.add_trace(go.Box(y=np.array(dfd6   ['Throughput_A']), name=6))
fig_A1.add_trace(go.Box(y=np.array(dfd7   ['Throughput_A']), name=7))
fig_A1.add_trace(go.Box(y=np.array(dfd8   ['Throughput_A']), name=8))
fig_A1.add_trace(go.Box(y=np.array(dfd9   ['Throughput_A']), name=9))
fig_A1.add_trace(go.Box(y=np.array(dfd10  ['Throughput_A']), name=10))

fig_A1.update_layout(xaxis_title="Distance between AP and Node A (m)",
                  yaxis_title="Node A Throughput (Mbps)")
fig_A1.show()
```