

**A NEW PROTECTION TECHNIQUE FOR  
DOMAIN NAME SYSTEMS BASED ON RAID TECHNOLOGY**

<b>Jawad Azam</b>	<b>201910688</b>
<b>Sara Al Halwani</b>	<b>201910372</b>
<b>Al Aiz Harhara</b>	<b>201730099</b>

**A project report submitted in partial fulfilment of the  
requirements for the award of the degree of  
Bachelor of Science in Cybersecurity**



**College of Engineering  
Al Ain University**

**20/04/2023**

## APPROVAL FOR SUBMISSION

I certify that this project report titled **“A NEW PROTECTION TECHNIQUE FOR DOMAIN NAME SYSTEMS BASED ON RAID TECHNOLOGY”** was prepared by **Jawad Azam, Sara Al Halwani, and Al Aiz Harhara** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science in Cybersecurity at Al Ain University, Al Ain.

Approved by,

Signature : \_\_\_\_\_

Supervisor: Dr.

Date : \_\_\_\_\_

## **A NEW PROTECTION TECHNIQUE FOR DOMAIN NAME SYSTEMS BASED ON RAID TECHNOLOGY**

### **ABSTRACT**

*DDoS attacks has been a major problem attack on DNS servers around the world for years, and yet no actual solution was found to deal with such an attack at any time. Therefore, our team came up with the idea of creating a DNS network that is based on the concept of RAID technology, where instead of relying on 1 DNS server, we will be working with 3 DNS servers that does not even share the same records, instead it shares part of the main records, where if one of the servers gets attacked or damaged; the other two can still perform and get the job done, and without losing data or time in the process.*

## TABLE OF CONTENTS

<b>APPROVAL FOR SUBMISSION</b>	<b>2</b>
<b>ABSTRACT</b>	<b>3</b>
<b>TABLE OF CONTENTS</b>	<b>4</b>
<b>LIST OF FIGURES</b>	<b>6</b>
<b>CH1: INTRODUCTION</b>	<b>7</b>
<hr/>	
<b>1.1 PROJECT OVERVIEW</b>	<b>7</b>
<b>1.2 PROJECT SCOPE</b>	<b>8</b>
<b>1.3 PROBLEM STATEMENT</b>	<b>9</b>
<b>1.4 AIMS AND OBJECTIVES</b>	<b>10</b>
<b>1.5 PROJECT MANAGEMENT</b>	<b>11</b>
1.5.1 TASK DISTRIBUTION	11
1.5.2 PROJECT PLAN (GANTT CHART)	12
<b>CH2: BACKGROUND AND LITERATURE REVIEW</b>	<b>13</b>
<hr/>	
<b>2.0 DOMAIN NAME SYSTEM (DNS)</b>	<b>13</b>
<b>2.1 WHAT SECURITY DOES DNS HAVE?</b>	<b>14</b>
<b>2.2 WHAT SECURITY DOESN'T DNS HAVE?</b>	<b>16</b>
<b>2.3 REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID)</b>	<b>16</b>
2.3.1 TYPES OF RAIDS	16
2.3.2 RELIABILITY ACHIEVED BY RAID	17
2.3.3 RELATED WORK (RAID 5)	18

<b>CH3: DESIGN METHODOLOGY</b>	<b>19</b>
<b>3.1 DESIGN SPECIFICATIONS</b>	<b>19</b>
<b>3.2 DESIGN CONSTRAINTS</b>	<b>21</b>
<b>3.4 PROPOSED DESIGN</b>	<b>23</b>
3.4.1 DECISION MATRIX	23
3.4.2 COMPONENTS	24
3.5 DESIGN IMPROVEMENT	27
<b>CH4: ENGINEERING CONSIDERATIONS</b>	<b>28</b>
<b>4.1 ENGINEERING ETHICS</b>	<b>28</b>
<b>4.2 IMPACT OF PROPOSED SOLUTION</b>	<b>30</b>
<b>CH5: RESULTS AND DISCUSSION</b>	<b>31</b>
<b>5.1 IMPLEMENTATION AND TESTING</b>	<b>31</b>
<b>5.2 RESULTS AND PERFORMANCE EVALUATION</b>	<b>35</b>
<b>CH6: CONCLUSION AND FUTURE WORK</b>	<b>42</b>
<b>REFERENCES</b>	<b>43</b>

## LIST OF FIGURES

<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
1	DDoS attack frequency compared with peak DDoS attack sizes, January 2020 through March 2021	7
2	DNS main components and functionality process.	9
3	RAID 5 Functionality, Hard disks and parity blocks.	14
4	RAID technology Concept	18
5	Domain Name Systems based on RAID technology	19
6	Total delay time of different storage size compared to a basic DNS.	36
7	Total delay time of different storage size using 3 storages and 4 compared to a basic DNS.	36
8	Total delay time of different storage size using 3 storages and 4 compared to a basic DNS during a DDoS attack using High Orbit Ion Cannon (HOIC) on one of their storages.	38

## CHAPTER 1

# Introduction

### 1.1 Project Overview

The problem that the project aims to solve is the vulnerability of Domain Name Systems (DNS) to Distributed Denial of Service (DDoS) attacks. DDoS attacks involve overwhelming a DNS server with traffic, making it difficult or impossible for legitimate users to access the website or service associated with that server. This type of attack can cause significant disruptions to internet services and can also be used as a tool for cybercriminals to extort money from website owners.

The proposed solution is to use a network of three DNS servers based on RAID (Redundant Array of Independent Disks) technology. RAID is a method of storing data on multiple hard drives to increase reliability and performance. In the proposed DNS network, the three servers would not share the same records but instead would hold different parts of the main records. This setup ensures that if one server is attacked or damaged, the other two can still function and complete the task without losing data or time.

The project aims to create a more robust and reliable DNS system that can withstand DDoS attacks. The proposed solution offers several benefits, including increased resilience and availability of the DNS network, improved response times, and enhanced security. By using RAID technology and distributing the DNS records among multiple servers, the proposed solution can reduce the impact of DDoS attacks and ensure that internet services remain functional even under attack.

The project would likely involve developing a custom DNS software that can distribute DNS records across multiple servers and manage requests and responses. Additionally, it would require setting up and configuring the three DNS servers in a way that ensures redundancy and fault tolerance. The project would also require extensive testing to ensure that the proposed solution is effective and reliable under various conditions.

In conclusion, the project offers a promising solution to a long-standing problem of DNS vulnerability to DDoS attacks. By leveraging RAID technology and distributing DNS records across multiple servers, the proposed solution can provide a more resilient and reliable DNS system that can withstand DDoS attacks and improve internet security and stability.

## 1.2 Project Scope

- i. Research and analysis of existing DNS server and storage server implementations to understand the problem that the project aims to solve.
- ii. Development of custom DNS and storage server software that can communicate using UDP sockets.
- iii. Selection of appropriate hardware and software tools to set up and configure the DNS and storage servers.
- iv. Implementation of the proposed solution, including the installation and configuration of the DNS and storage servers and the custom software.
- v. Extensive testing of the proposed solution to ensure that it is effective and reliable under various conditions, including simulated network congestion and server failures.
- vi. Documentation of the project, including detailed design and implementation documents, test results, and user manuals.



### 1.3 Problem Statement

DNS servers are the main storage for all the IP addresses in the web, and if an attacker gets his hands on the server, then he will have access to all the IP addresses. Additionally, losing all the DNS data through destruction or theft due to having all the data stored in one place is a huge risk. Companies can lose a lot either financially or physically if they're using a private DNS server and its data gets corrupted or destroyed without a way back.

The only current solution for such a vulnerability is creating backups of the DNS server data, but how reliable is this solution? Once a DNS server gets attacked and loses the data it contains, recovering and loading the backed-up data is not instant and takes time, the more seconds passed having the DNS server down recovering, the bigger the loss becomes.

Figure 1 contrasts the monthly average for DDoS assaults (purple bars) with the month's biggest attack (yellow line). Although we notice a slow but continuous increase month after month, the number of attacks stays basically stable. In reality, the number of monthly attacks increased by 62% between January 2020 and March 2021. [7]

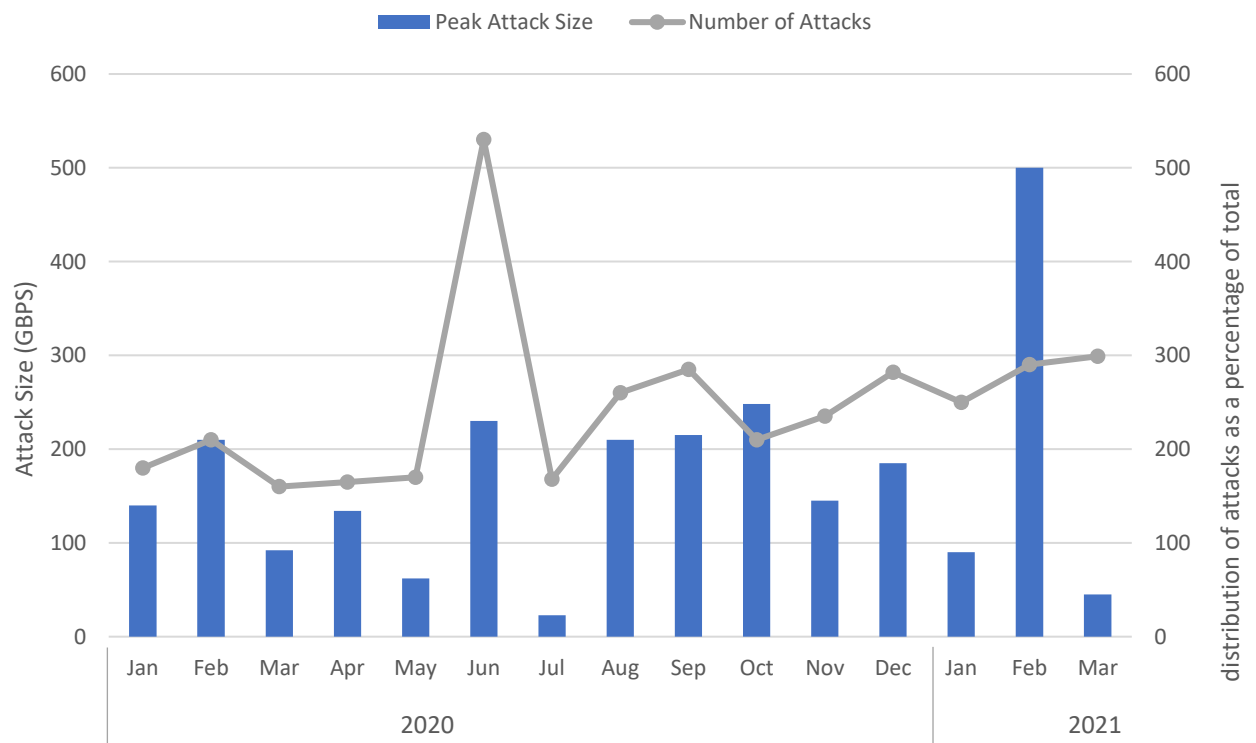


Figure 1. DDoS attack frequency compared with peak DDoS attack sizes, January 2020 through March 2021 [7]

## 1.4 Aims and Objectives

### The Project Aims:

- i. The project aims to verify a new method/technique to decrease the negative effects of DDoS attacks on DNS servers or the effects of DNS flooding.
- ii. Therefore, the project mainly aims to combine DNS server with RAID technology to make the DNS server storages protected and ready to face any attack that could damage the data or disturbed its functionality.

### The Project Objectives:

- To search & investigate deeply into the DNSSEC topic, explore its weaknesses & strength, as well as how it works and how it's being protective.
- To clearly describe what RAID technology is, what are RAID types and the difference between them.
- Design the method/technique that we will use to combine the DNS server with RAID technology.
- Implement & test the design as well as recording results of performance and other criterias to be considered, along with formulating a result and a performance evaluation.
- Apply the IEEE code of ethics in our project & discuss its impact on the world.







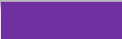
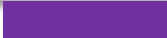





## 1.5 Project Management

### 1.5.1 Task Distribution

The distribution of the project tasks between the team members was as follows:

Members \ Tasks	Jawad Azam	Sara Al Halwani	Al Aiz Harhara
Introduction			
Collecting information for the literature			
Writing the literature			
Design Specifications			
Design constrains			
Describing the Proposed Design			
Engineering Ethics & Impact of Proposed Solution			
Results and discussion chapter			
Writing The Project Conclusion			

### 1.5.2 Project Plan (Gantt chart)

Tasks	Start	End	Duration	Spetember 2022	Ocrober 2022	November 2022
Formulating Project Team & Defining the problem	13-Sep	15-Sep	2 Days			
Chapter 1: Introduction						
1.1 Problem Statemen Along with Aims and Objectives	15-Sep	19-Sep	4 Days			
Chapter 2: Background Literature						
2.1 Search & Prepare Articles, Books & Journeys	19-Sep	22-Sep	3 Day			
2.2 Review on Similar Projects & Literatures	22-Sep	29-Sep	7 Day			
2.3 Writing the Literature Review	29-Sep	13-Oct	14 Day			
Chapter 3: Design Methodology						
3.1 Design Specifications	15-Oct	20-Oct	5 Days			
3.2 Design Constraints & Alternative Designs	20-Oct	22-Oct	2 Days			
3.3 Describing the Proposed Design & future Improvements	27-Oct	13-Nov	17 Days			
Chapter 4 & 5: Engineering Considerations & 5 Project Management						
4.1 & 2 Engineering Ethics & Impact of Proposed Solution	13-Nov	15-Nov	2 Days			
4.3 & 5.1 Engineering Standard & Project Plan	15-Nov	17-Nov	2 Days			
Chapter 7: Conclusion & Final Report						
7.1 Writing The Project Conclusion	17-Nov	18-Nov	1 Day			
7.2 Formulate The Project Final Report	18-Nov	24-Nov	6 Days			
Project Presentation	29-Nov	29-Nov	1 Days			

## CHAPTER 2

# Background And Literature Review

To identify machines that may be reached via the Internet or other IP networks, a hierarchical and decentralised naming system called the Domain Name System (DNS) is employed, [1][13] and so it works as the library of the IP addresses on the internet. The DNS servers if not secured properly and get attacked; it could result in leakage of the server's data, disables the server's service, or the server could completely lose its data.[14] And yet in our world, beside creating backups for the DNS records, there is no actual mechanism used to recover lost data in DNS servers.[15]

### Domain Name System (DNS)

The DNS server's main functionality is to translate the IP addresses to websites and vice versa, but how does that process actually work? To start off, there are 4 types of DNS server[16]:

- DNS recursive resolver / DNS resolver
- Root name server
- Top level domain / TLD name server
- Authoritative name server

The DNS resolver's main functionality is to connect the browser of a computer to the rest of the DNS server. There are 13 sets of root name servers named 'letter.root-servers.net'. TLD name server stores the information of the URL domains such as: .com, .net, .org, etc. Authoritative name server is where the actual IP addresses are stored inside the DNS[16][17].

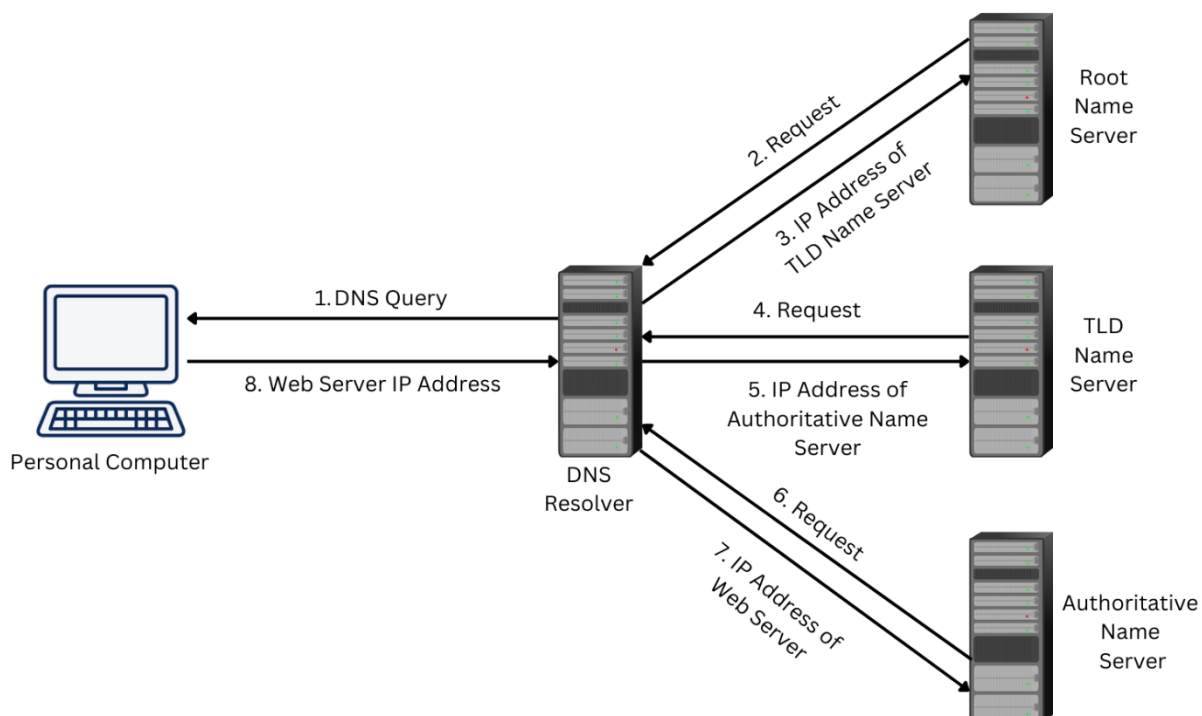


Figure 2. DNS main components and functionality process.

The process inside the DNS server starts when the user types a certain URL in his browser, then the browser will send the URL to the operating system and the operating system sends a query to the DNS resolver asking for the IP address for the URL as shown in figure 2. Next up, the DNS resolver will check if it contains the IP address for the requested website in its cache, if not then the DNS resolver will send a request to the root name server and receive the IP address of the TLD name server. The DNS resolver will once again then request the TLD name server and receive the IP address of the authoritative name server and receive the IP address of the web server requested. Once the DNS resolver gets the IP address of the web server it will cache it for future use and send it to the operating system in the personal computer, where the operating system will hand it to the browser[16][17][18].

## 2.1 What security does DNS have?

The Domain Name System (DNS) security extensions include means for validated denial of DNS data's existence in addition to origin authentication and data integrity assurance services. Below is a description of these mechanisms[2][3][8].

The DNS protocol must be modified to accommodate these approaches. Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS), and Next Secure are the four new resource record types added by DNSSEC (NSEC). Additionally, it adds the Checking Disabled (CD) and Authenticated Data message header flags (AD).[8] Additionally, DNSSEC needs EDNS0 support to support the bigger DNS message sizes brought on by the addition of the DNSSEC RRs. Finally, for a security-conscious resolver to specify in its queries that it intends to receive DNSSEC RRs in response messages, DNSSEC requires support for the DNSSEC OK (DO) EDNS header bit. These services shield the Domain Name System from most threats[2][8].

Key distribution, data origin authentication, and transaction and request authentication are the three independent services offered by the Domain Name System (DNS) protocol security extensions.[10]

### **Key Distribution**

For the purpose of connecting keys to DNS names, a resource record format is defined. This enables the DNS to support other protocols as well as DNS security by acting as a public key distribution mechanism.[8]

An algorithm identifier, the actual public key parameter(s), and several flags, including those identifying the type of entity the key is associated with and/or stating that there is no key associated with that entity, are all included in the syntax of a KEY resource record (RR). To reduce the number of queries required, DNS servers that are security conscious will automatically try to return KEY resources as supplementary information in addition to those resource records that were really requested[2][3].

### **Data Origin Authentication and Integrity**

Associating with resource record sets (RRsets) in the DNS cryptographically generated digital signatures provides authentication. Typically, a single private key will be used to authenticate a complete zone, although there may be several keys for various algorithms, signers, etc. For signed data read from that zone, a security-aware resolver can securely validate that it is authorised by learning the zone's public key.[11] The zone private key(s) should be stored offline and regularly used to re-sign all the zone's records for the most secure implementation. However, there are situations in which DNS private keys must be online, such as dynamic update[3][10].

### **DNS Transaction and Request Authentication**

The resource records that can be retrieved and their absence are both protected by the data origin authentication service discussed above, but neither DNS requests nor message headers are covered.[3]

There isn't much that can be done if a poor server sets header bits incorrectly. However, transaction authentication can be added. A resolver can be confident that it is at least receiving messages from the server it believes it contacted and that the response is in response to the query it sent thanks to such authentication (i.e., that these messages have not been doodled in transit)[9]. This is achieved by opportunistically appending a special SIG resource record that digitally signs the union of the resolver's query and the server response at the end of the reply[3][8][9].

## 2.2 What security doesn't DNS have?

The DNS was initially built on the presumption that all data in the DNS is thus visible, and that the DNS will always respond the same way to any given query, regardless of who may have issued it. Therefore, secrecy, access control lists, or other methods of discriminating amongst inquirers are not intended to be provided by DNSSEC[8].

No defence against denial-of-service assaults is offered by DNSSEC. Resolvers and name servers that are security conscious are susceptible to a different category of denial-of-service attacks that use cryptographic processes.[2][3]

## 2.3 Redundant Array of Independent Disks (RAID)

By storing the same data across many hard disks or solid-state drives, RAID (redundant array of independent disks) is a method for protecting data in the case of a drive failure (SSDs). There are many RAID levels, though, and not all of them strive to offer redundancy[19].

### 2.3.1 Types of RAIDS

Different versions, known as levels, are used by RAID devices. The six levels of RAID — 0 through 6 — were specified in the original paper that created the term and the RAID configuration concept. These numbers made it possible for IT professionals to distinguish between RAID versions. Since then, the number of levels has increased and been divided into three groups: regular, nested, and non-regular RAID levels[20][21].

Level 3 RAID 5: This level consists of at least three storage disks spread throughout several disks. as this level combines the Parity method of dealing with data as a single block, and reliance on part of the first disk method Striping[21], which is the high reading speed. This level is characterised by the continuous operation of the matrix until one of the disks falls or stops working.[24] Where this level is considered one of the levels commonly used in organisations that produce several data that cannot be afforded to lose.[6][26][24]

Level 4 RAID 6: It uses block-level striping with two parity blocks dispersed across all member drives because it extends RAID 5 by adding an additional parity block. Like RAID 5, RAID 6 disk arrays can be configured in a variety of ways depending on the way the data blocks are written, where the parity blocks are placed in relation to the data blocks, and whether the first data block of a subsequent stripe is written to the same drive as the last parity block of the prior stripe. [23][24]



### 2.3.2 Reliability Achieved by RAID

RAID's redundancy is a fault-tolerant technology used to address the issue of data loss due to failed disk drives. Disk arrays also provide a substantial advantage over traditional disks.[22]

We must employ an additional disk carrying redundant data to retrieve the original information in the event of a disk failure in order to address the dependability challenge.[5] These Redundant Arrays of Inexpensive Disks are known as RAID. We provide a taxonomy of five distinct groupings of disk arrays, starting with mirrored disks and moving through a variety of alternatives with varying performance and reliability,[4] in order to clarify the explanation of our final solution and avoid confusion with prior work. Each organisation is referred to as a RAID level. The reader should be reminded that we only describe all levels as though they were implemented in hardware to make the presentation simpler. RAID notions apply to both hardware and software implementations. Reliability.[4][5][22]

### 2.3.3 Related Work (RAID 5)

RAID 5 technology is the technology that we will be using initially for our design, but first let's talk about how it functions. In figure 3, we can see that in our RAID 5 storage system here, we have 3 hard disks containing 3 different data (A, B, C). Each data is divided in half and stored in 2 different hard disks, while keeping a parity of that data in another separate disk. If one of the hard disks fails or goes down, the system would still work considering that the data from the failed hard disk can still be retrieved easily and the system would not have an issue to completely stop.[24]

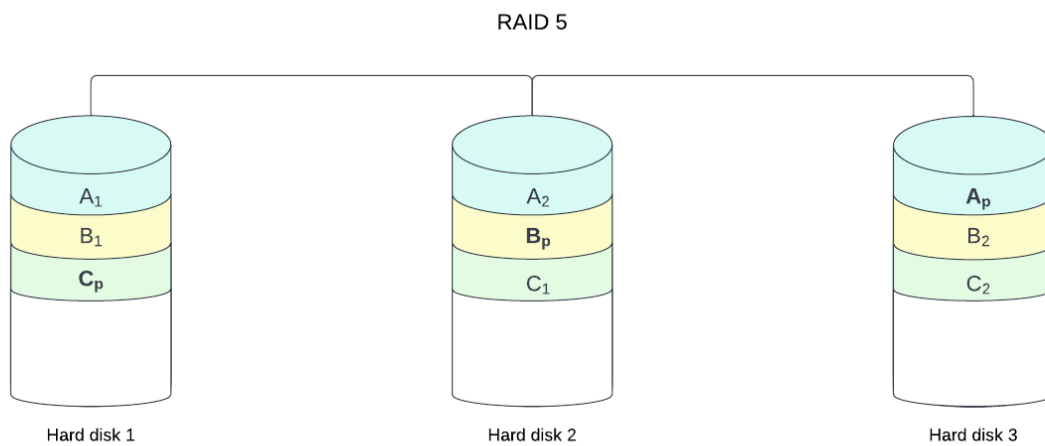


Figure 3. RAID 5 Functionality, Hard disks and parity blocks.

Let's say for example that A = 0110 0010, the first 4 bits will be combined with the second 4 bits and it will be divided into Hard disk 1 and Hard disk 2 as follows:

A<sub>1</sub> = 0110 → Hard disk 1

A<sub>2</sub> = 0010 → Hard disk 2

A<sub>p</sub> = 0100 → Hard disk 3

A<sub>p</sub> which is the parity for A, it will be calculated using X-OR,

$$\text{so } A_1 \oplus A_2 = A_p$$

And so on for B and C as well. Therefore, if one of the hard disks gets destroyed for example, just by reversing the calculations we can obtain the lost hard disk data

## CHAPTER 3

# Design Methodology

### 3.1 Design Specifications

Our capstone project idea is completely based in the digital world, and therefore, beside a personal computer, our design will not require any other physical materials to be available. Thus, we will be working with a simple software that has the ability to create virtual machines for us.

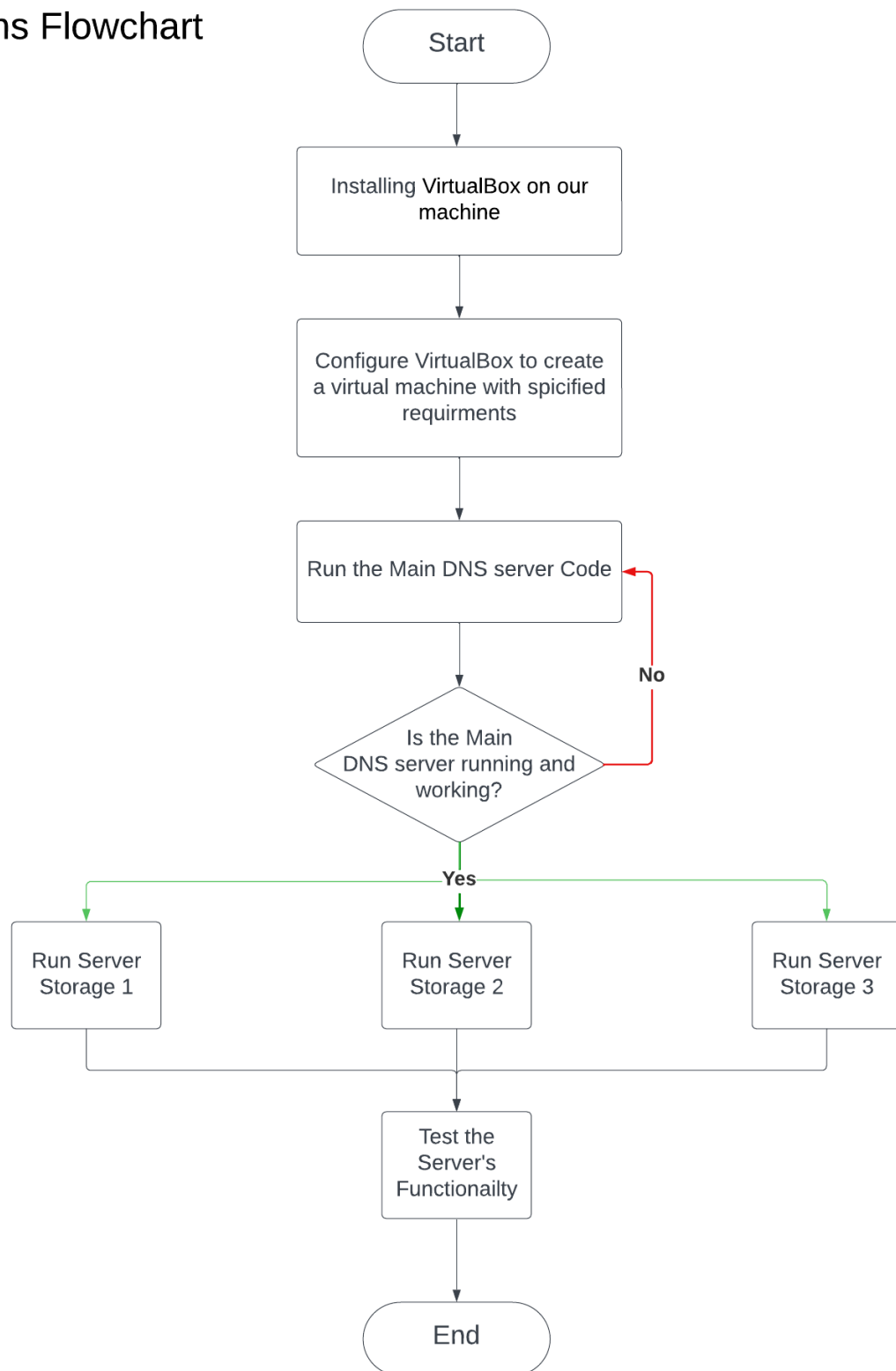
The software will take some of our personal computer resources and create two independent virtual machines which have the ability to fully virtualize a computer system for us, where we can implement our design idea in.

VirtualBox is what the software that we'll be using is called, it was developed by oracle corporation in 2010. It's a type 2 hypervisor which means that it's going to be installed on our existing operating system, which means reinstalling our operating system to run a virtual machine won't be required.

We will be creating 1 virtual machine, giving it the required operating system, memory size, and number of processors to be able perform as our DNS server host machine. Once the virtual machine gets sat up successfully, we will need to install and configure a public/private DNS server on it, and we will call that machine DNS Machine.

VIRTUAL MACHINE CHARACTERISTICS	
Operating System	Ubuntu
Base Memory	4 GB
Video Memory	128 MB
Hard Disk Size	200GB
Processors	2 CPUs

## Specifications Flowchart



## 3.2 Design Constraints

### Performance

In the normal cases, when a PC user types the website URL and hits enter, the computer will directly send a request to the DNS server for the IP address of that URL and return the results. Setting up a RAID storage means that the DNS server now will have to deal with more than 1 storage, and according to our design it will have to send the request to 3 storages instead of 1. In addition, receiving 3 replies from all the storages is a must.

Therefore, more steps are going to be required in the process, resulting in affecting the performance negatively. Our group will examine and look into the performance of our design in-depth in the implementation phase.

### Cost

The system should be designed to be cost-effective, using affordable hardware and software components where possible, and when it comes to cost, there are a few factors to consider such as the hardware, software, networking, maintenance and support.

For instance, CPU, the required processing power of the CPU will depend on various factors. Higher-end CPUs with more cores and faster clock speeds will generally be more expensive than lower-end CPUs. The cost of the CPU will likely be one of the largest hardware costs associated with the project. Regarding the memory, the amount of memory required will depend on the size of the data being stored and retrieved, as well as the number of clients making requests. Higher memory capacities will generally be more expensive than lower capacities.

Also, the storage servers will require hard drives or solid-state drives (SSDs) to store the data being retrieved. The cost of the storage will depend on the amount of data being stored and the desired level of redundancy (e.g., RAID 1, RAID 5, etc.). Higher-capacity drives will generally be more expensive than lower-capacity drives, and SSDs will generally be more expensive than hard drives. Along with the networking, where the DNS and storage servers will need to be connected to each other and to clients via a network infrastructure. The cost of the networking equipment will depend on the size and complexity of the network, as well as the desired level of redundancy and performance. Higher-quality networking equipment will generally be more expensive than lower-quality equipment.

## Reliability

Reliability is a critical design constraint for this project, as the DNS and storage servers will be responsible for handling and storing important data that must be available to clients at all times. To ensure high reliability, the following measures can be taken:

One way to increase reliability is to implement redundancy, which means having backup systems that can take over if the primary system fails. In the case of this project, redundancy can be achieved by using multiple DNS and storage servers, as described earlier. Redundancy can also be achieved by using redundant power supplies, network connections, and storage devices.

In addition, Fault tolerance refers to a system's ability to continue operating even if some of its components fail. To achieve fault tolerance, the DNS and storage servers can be designed with mechanisms to detect and recover from failures. For example, if one of the storage servers fails, the system can automatically switch to another server to retrieve the data.

## Compatibility with UDP

Compatibility with UDP is a design constraint because the DNS server and storage servers in this project communicate with each other using UDP sockets. And to ensure compatibility with UDP, we need to be considering couple measures such as the protocol compliance, where the UDP protocol has certain rules and specifications that must be followed for communication to be successful.

We also need to be considering the network topology, since UDP is a connectionless protocol, it means that packets can be sent and received without establishing a connection first. However, the network topology can affect the reliability of UDP communication. For example, if there are firewalls or NAT devices between the DNS server and storage servers, they may need to be configured to allow UDP traffic to pass through.

Lastly, we need to consider the port availability, because UDP uses port numbers to identify different applications and services. And to ensure compatibility with UDP, the DNS server and storage servers must use available and appropriate port numbers.

### 3.4 Proposed Design

#### 3.4.1 Decision matrix

	RAID 5	RAID 6	RAID 10
Selection criteria	Value (V)	Value (V)	Value (V)
Minimum number of drives	3	4	4
Protection	Parity protection for single disk failure	Parity protection for two disk failures	Mirror protection
Fault tolerance	Single-drive failure	Two-drive failure	Up to one disk failure in each sub-array
Read performance	Low	Low	High
Write Performance	Low	Low	Medium
Capacity utilisation	67% – 94%	50% – 88%	50%

There are multiple types of RAID technology, and they all have their strengths, weaknesses, and specialties. However, the team was aiming on 3 main types of RAID technology, which are RAID 5, 6, and 10.

For our initial design, we will be focusing on using RAID 5, since the least required hard disks drivers are 3 which can be easily afforded. It has a lower read & write performance than the other 2 types, but its structure is easier and simpler to design and combine with DNS servers than RAID 6 or 10.

However, it has tolerance for a single disk failure only, but if the design worked out and showed good results in the implementation phase, the design can be upgraded and improved to be working with RAID 6 or 10, which in case of RAID 6 it has parity protection for two disk failures for example.

### 3.4.2 Components

#### *Concept*

Generally, our design will function using the RAID concept as follows: There will be 3 main components: the Personal Computer (PC), the DNS server, and the Main RAID Storage as shown in figure 4. The PC will send a request R to the DNS server requesting for the IP address for website R. The DNS server will then send this request R to the main storage (it's Database), which is going to be sat up as a RAID storage, to retrieve the information required.

As mentioned before, in our design the main storage will not be in one part or in one place, rather it will be divided within many parts after being set up as a RAID storage which is going to be 3 different DNS servers. For example, if we are planning to use the concept of RAID 5, the storage will be divided to minimally 3 parts within the main storage, so will be using 3 DNS server; to be able to perform recovery from a failure of a single drive/server. After that, the DNS server will send 3 identical requests R, these requests should be all sent at the same time to the 3 different storage parts (S1, S2, S3) inside the RAID 5 storage, but in the main design, instead of the 3 RAID storages, 3 DNS servers will play in place.

Afterwards, since the information requested will be previously divided and saved in two different storage parts of the 3, the RAID 5 storage will look for the request R in all of its storages. Once the information needed is found, collected, and connected, it will be retrieved by the DNS server and sent to the personal computer as the result of its R request.

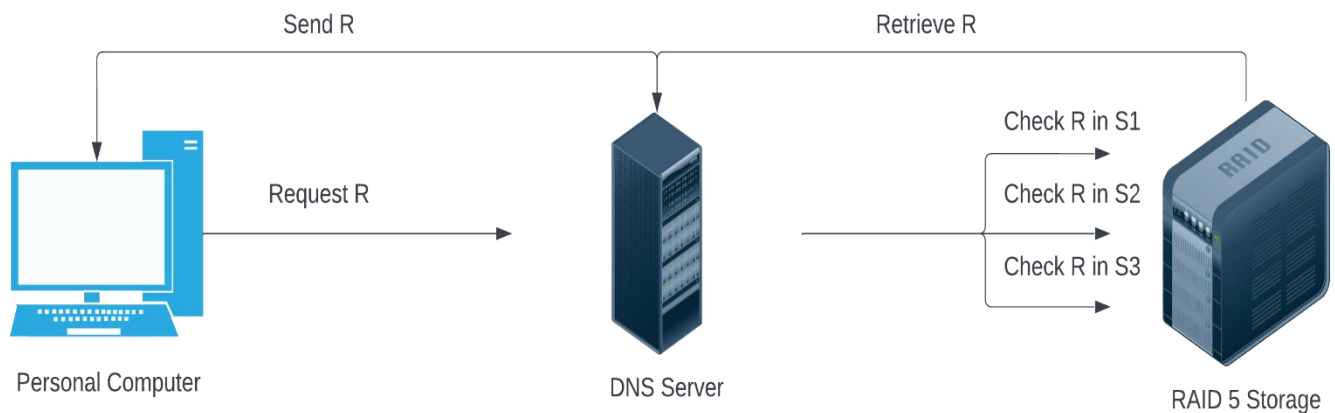


Figure 4. RAID technology Concept



### *Proposed Design*

By applying the RAID 5 concept in our design, we will have a DNS server that can still perform under different types of attacks such as DDoS. As shown in figure 5, the DNS records of text format will be converted to binary format and stored inside the 3 different DNS servers. The binary format will be divided into two sets, one set in each DNS server storage and the result of an XOR between the two sets will be stored in the last DNS server storage. SO, when a user types in the URL of a website, it won't directly be sent to the DNS server as a request to retrieve the URL IP address, instead it will be going through a process that is going to: first, convert the URL from text format to binary format, in order to make storing it easier. Second, the binary numbers will be divided into half, and XOR the two halves of the binary number, as a result; we will end up with 3 binary set of numbers generated through this process. Once that is done, the 3 binary set of numbers will be sent as a request to the 3 DNS servers. The request is expected to be sent at the same time, and so as the results retrieval it should be sent back at the same time.

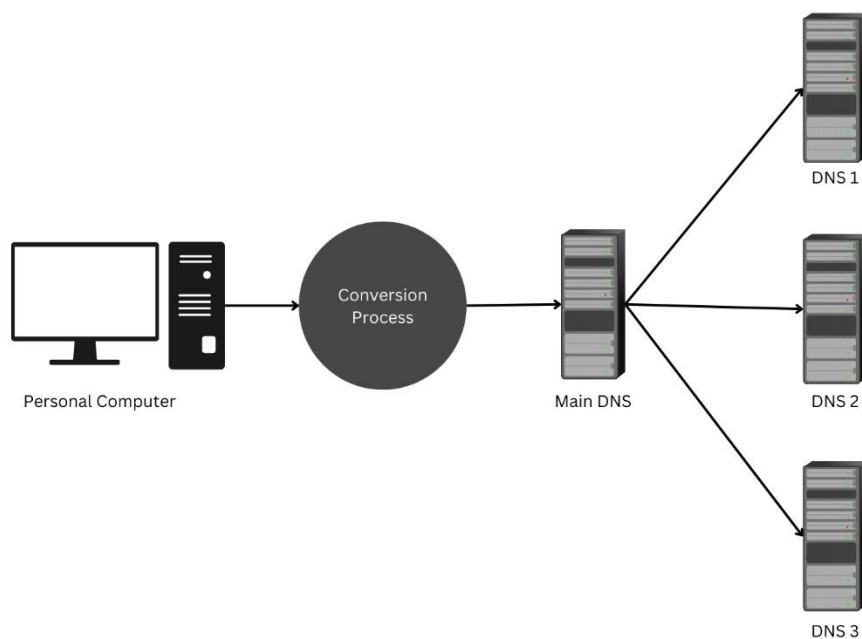


Figure 5. Domain Name Systems based on RAID technology

The pseudocode of the conversion process will be as follows:

```

1:      Calculate length of string
2:      for (i<length) do
3:          Convert Char to ASCII
4:          Convert ASCII to Binary
5:      end for
6:      print output

```

For converting Char to ASCII inside the for loop:

```

7:      set val to Integer value of (s.charAt(i))

```

For converting ASCII to Binary:

```

8:      set x equals to empty string
9:      while val > 0 do
10:         if val % 2 == 1 then
11:             x += '1'
12:         else
13:             x += '0'
14:         val /= 2
15:     end if

```

DNS request pseudocode:

```

16:     scan request x
17:     Set request r = x.ConvertToBinary
18:     r = r.length/2
19:     Set r1 = first half of binary set numbers
20:     Set r2 = second half of binary set numbers
21:     Set r3 = r1 XOR r2
22:     send requests r1, r2, r3 to DNS1, DNS2, DNS3
23:     if (results found) do
24:         Return results
25:     else
26:         Return no results found
27:     end if

```

### 3.5 Design Improvement

The design has a huge potential for improvements, after the implementation and testing phase, if the design performed well and had almost no issues; the team looks forward to upgrading and improving the design by implementing RAID 6 technology into the design instead of RAID 5.

With such an addition, the design functionality, speed, performance, and more, would easily get improved a lot more. Along with the new speciality to having a parity protection for two disk failures instead of one.

Along with couple potential design improvements that can be considered which are mentioned below:

#### **Load balancing**

While the use of three storage servers provides redundancy and fault tolerance, there is no load balancing mechanism in place to distribute the workload across the servers. As a result, one server may become overloaded while the others are underutilized. To address this, a load balancing mechanism could be implemented to evenly distribute requests across all three servers, which would help improve overall system performance.

#### **Security**

The current design does not address security concerns such as data encryption and authentication. To improve security, data encryption could be implemented to protect against data breaches or interception during transmission. Additionally, user authentication could be added to prevent unauthorized access to the DNS and storage servers.

#### **Scalability**

The current design is limited to three storage servers, which may not be sufficient for larger-scale deployments. To improve scalability, the system could be designed to support additional storage servers as needed, with automatic load balancing to ensure even distribution of requests.

#### **Monitoring and logging**

To improve system management and troubleshooting, it may be beneficial to implement monitoring and logging capabilities. This would enable administrators to monitor system performance, detect issues, and troubleshoot problems as they arise.

## CHAPTER 4

# Engineering Considerations

### 4.1 Engineering Ethics

Keeping the project ethical and legal is one of the major parts that the team focuses on, because it represents the team's honesty and responsibility, and works as a protection for the project in case of conflicts of thoughts that might occur.

The IEEE code of ethics were applied in our project, where the team made sure to avoid unlawful conduct in our work and activities. As well avoiding the injuring of others, or their property, or reputation, or employment by false or malicious actions. Also, not engaging in any harassment of any kind. And not discriminate against anyone based on their race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression. treat all people equally and with respect. While also maintaining and improving our technical competence, and to only perform technological tasks for others if they are trained, experienced, or after being fully informed of any relevant limitations.

The Domain Name System (DNS) is a critical component of the internet, allowing users to access websites and other online resources using human-readable domain names rather than IP addresses. However, DNS servers are vulnerable to Distributed Denial of Service (DDoS) attacks, which can cause significant disruptions to internet services. One potential solution to this problem is a DNS system based on RAID technology, which uses multiple servers to provide redundancy and improve reliability. However, designing and implementing such a system requires careful consideration of engineering ethics considerations to ensure that the system is safe, reliable, and ethically responsible.

#### Responsibility to Public Safety:

One of the key engineering ethics considerations for a DNS system based on RAID technology is the responsibility to public safety. Because DNS servers are critical components of the internet infrastructure, any failures or vulnerabilities in the system could have significant consequences for public safety. Therefore, it is important to design and implement the system in a way that prioritizes reliability, security, and safety. This could include implementing measures to prevent DDoS attacks, ensuring that the system is resilient to hardware failures, and testing the system thoroughly to identify and address any potential safety risks.

#### Responsibility to the Environment:

Another important engineering ethics consideration for the DNS system based on RAID technology is the responsibility to the environment. The design and implementation of the system could have environmental impacts, such as through the energy consumption of the servers or the materials used in their construction. Engineers should consider ways to minimize these impacts, such as by using energy-efficient hardware or implementing recycling programs for old hardware. Additionally, the system should be designed with scalability in mind to minimize the need for additional hardware in the future.

#### Ethical Use of Technology:

Engineers also have a responsibility to ensure that the DNS system based on RAID technology is used ethically and does not contribute to harm or injustice. This could include designing the system with privacy and security in mind, to prevent unauthorized access to user data or DNS records. Additionally, the system should be designed to prevent misuse for illegal or unethical purposes, such as by implementing measures to detect and prevent DNS-based attacks or cybercrime.

#### Professionalism and Integrity:

Finally, engineers have a responsibility to act with professionalism and integrity in their work. This could include being transparent about any limitations or potential risks associated with the system, and avoiding conflicts of interest or other behaviours that could compromise the integrity of the work. Additionally, engineers should prioritize collaboration and open communication to ensure that the system is designed and implemented in a way that meets the needs of all stakeholders.

## 4.2 Impact of Proposed Solution

Our solution completely works and based in the digital world, therefore it has no impact what so ever on the earth health and the environment. However, if the solution makes a huge success and works perfectly, it is going to affect the global world economically in a positive way, where companies can save themselves millions of dollars loses that happens because of DDoS attacks through implementing our idea/solution.

The proposed solution involves using three DNS servers that do not share the same records, which can improve reliability by ensuring that DNS services remain available even if one server is attacked or damaged. Furthermore, this approach can help to mitigate the impact of DDoS attacks, which can have a major impact on DNS services. With multiple DNS servers working together, the impact of an attack on any single server can be reduced, making it more difficult for attackers to take down DNS services.

Additionally, the use of RAID technology in the proposed solution can improve the integrity of data stored in DNS servers. By spreading data across multiple servers, the risk of data loss due to hardware failure or other issues can be reduced. Furthermore, this approach can lead to better performance by distributing requests across multiple servers, resulting in faster response times for DNS queries.

In conclusion, the proposed solution for DNS protection based on RAID technology has the potential to make a significant impact on DNS server reliability, security, and performance. By using multiple servers that do not share the same records, the impact of DDoS attacks can be reduced, data integrity can be improved, and performance can be enhanced. While there may be some design constraints, such as the cost of implementing additional hardware, the benefits of this approach could outweigh these challenges. Overall, this solution represents an important step forward in improving DNS services and protecting against attacks.

## CHAPTER 5

# Results And Discussion

### 5.1 Implementation and Testing

In this section, we describe the implementation process and testing methods used for our DNS server with RAID-5 methodology. We first implemented the DNS server in Python and then integrated the RAID-5 methodology to provide data redundancy and fault tolerance.

This code implements a simple DNS (Domain Name System) and storage servers. The DNS server is responsible for handling DNS requests and returning the corresponding IP address, while the storage servers are responsible for storing and returning data. The DNS server and storage servers communicate using UDP (User Datagram Protocol) sockets.

The DNS server listens on a specified IP address and port, and upon receiving a DNS request, it splits the request into two halves, XORs the two halves to create the third piece, and then sends requests to the three storage servers to retrieve the corresponding data. It checks which storage servers were used to retrieve the data and then performs reverse XOR operations to retrieve the original data. The storage servers listen on specified IP addresses and ports, and upon receiving a storage request, they retrieve the corresponding data and return it to the client. Finally, it converts the retrieved data to an IP address and sends it back to the client.

First, the data we are working with are stored in an external txt file called "storage\_data.txt" as raw data which are the websites with their corresponding IP address. Our code will start by reading the data from the txt file and then it will convert the data to binary, divide binary into two halves, XOR the two halves to create the third piece, and then stores each piece in a different DNS server storage.

The code includes the following functions:

- `handle_request(data)`: This function takes a DNS request in binary format as input, splits the request into two halves, XORs the two halves to create the third piece, sends requests to the three storage servers to retrieve the corresponding data, checks which storage servers were used to retrieve the data, performs reverse XOR operations to retrieve the original data, converts the retrieved data to an IP address, and returns the IP address.

- `serve_dns(ip, port)` function starts a DNS server that listens for DNS requests on a specified IP address and port. The function first creates a socket with the `socket.socket()` function and sets it to the IPv4 address family (`socket.AF_INET`) and the User Datagram Protocol (`socket.SOCK_DGRAM`). Then, it binds the socket to the specified IP address and port using the `bind()` method. The function then enters an infinite loop, waiting for incoming requests. When a request is received, it calls the `handle_request(data)` function to process the request and generate a response. Finally, it sends the response back to the client using the `sendto()` method of the socket. The function continues to listen for new requests until the server is shut down or encounters an error.

```

1. def serve_dns(ip, port):
2.     """
3.         Start the DNS server on the specified IP address and port.
4.     """
5.     with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as
server_socket:
6.         server_socket.bind((ip, port))
7.
8.         print(f"DNS server listening on {ip}:{port}...")
9.
10.        while True:
11.            # receive request
12.            data, address = server_socket.recvfrom(1024)
13.
14.            # handle request
15.            response = handle_request(data)
16.
17.            # send response
18.            server_socket.sendto(response, address)

```

- `send_request(address, data)`: function takes two parameters: `address`, which represents the address of the server to which the request is being sent, and `data`, which represents the data being sent in the request. The function uses the socket library to create a TCP/IP socket connection to the server at the specified address. It then sends the `data` parameter over the socket connection using the `sendall()` method. The function then waits for a response from the server using the `recv()` method.



```

1. def send_request(address, data):
2.     """
3.     Send a request to the specified DNS server and return the
       response.
4.     """
5.     with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as
       client_socket:
6.         client_socket.settimeout(5) # set timeout to 5 seconds
7.         client_socket.sendto(data.encode(), address)
8.
9.         try:
10.            response, _ = client_socket.recvfrom(1024)
11.        except socket.timeout:
12.            response = b""
13.
14.    return response

```

- `serve_storage(ip, port)` function starts a storage server that listens to incoming requests on the specified IP address and port. When a request is received, the function `handle_storage_request(request)` is called to handle the request and return the corresponding data. Currently, the `handle_storage_request(request)` function is not fully implemented, so it will return a `NotImplementedError` if the operation is not "get" or if the IP address is not in the `storage_data` dictionary. If the operation is "get" and the IP address is in the `storage_data` dictionary, then the corresponding data is returned in a dictionary with a "status" key set to "success" and a "data" key set to the data value.

```

1. def serve_storage(ip, port):
2.     """
3.     Start the storage server on the specified IP address and port.
4.     """
5.     with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as
       server_socket:
6.         server_socket.bind((ip, port))
7.
8.         print(f"Storage server listening on {ip}:{port}...")
9.
10.        while True:
11.            # receive request
12.            data, address = server_socket.recvfrom(1024)
13.
14.            # handle request
15.            response = handle_storage_request(data)
16.
17.            # send response
18.            server_socket.sendto(response, address)

```

- `handle_storage_request(request)`: is a function that handles a storage request and returns the corresponding data. It receives a dictionary request that has two keys: `ip_address` and `operation`. If the operation is "get", it checks whether the `ip_address` is in the `storage_data` dictionary. If it is, it returns a dictionary with two keys: "status" and "data", where "status" is set to "success", and "data" is set to the value of `ip_address` in the `storage_data` dictionary. If the `ip_address` is not in the `storage_data` dictionary, it returns nothing.

```

1. def handle_storage_request(request):
2.     """
3.     Handle a storage request and return the corresponding data.
4.     """
5.     ip_address = request["ip_address"]
6.     operation = request["operation"]
7.     if operation == "get":
8.         if ip_address in storage_data:
9.             data = storage_data[ip_address]
10.            return {"status": "success", "data": data}
11.        else:
12.            return {"status": "error", "message": "IP address not found
13.                                                    in storage."}
14.    else:
15.        return {"status": "error", "message": "Invalid operation."}

```

To test the functionality of the system, we used a combination of manual testing and automated testing using unit tests. We tested the system with a range of inputs and requests, including both valid and invalid requests. We also conducted stress testing to verify the system's performance under high load and DDoS attacks.

## 5.2 Results and Performance Evaluation

We present the results and performance evaluation of our DNS server with RAID-5 methodology. We first evaluated the system's performance under normal operating conditions and then tested its response to various DDoS attacks and DNS flooding.

We start by running the code on 5 different terminals where we allocate one for starting the DNS server, 3 for our 3 different DNS storages, and one for sending a DNS query (request)

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/JAWAD/Desktop/DNS Code/DNS_MainCode.py',
wdir='C:/Users/JAWAD/Desktop/DNS Code')
DNS server listening on 127.0.0.1:5000...
```

Starting serve\_dns

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/JAWAD/Desktop/DNS Code/
DNS_StorageCode.py', wdir='C:/Users/JAWAD/Desktop/DNS Code')
Storage server listening on 127.0.0.1:5001...
```

Starting serve\_storage 1

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/JAWAD/Desktop/DNS Code/untitled0.py',
wdir='C:/Users/JAWAD/Desktop/DNS Code')
Storage server listening on 127.0.0.1:5002...
```

Starting serve\_storage 2

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/JAWAD/Desktop/DNS Code/untitled1.py',
wdir='C:/Users/JAWAD/Desktop/DNS Code')
Storage server listening on 127.0.0.1:5003...
```

Starting serve\_storage 3

```
In [4]: runfile('C:/Users/JAWAD/Desktop/DNS Code/
DNS_FunctionalityCode.py', wdir='C:/Users/JAWAD/Desktop/DNS Code')
Test successful.. Requested {expected_response}, and received
176.32.108.166
Total time: 0.94187 Second

In [5]:
```

Testing Results

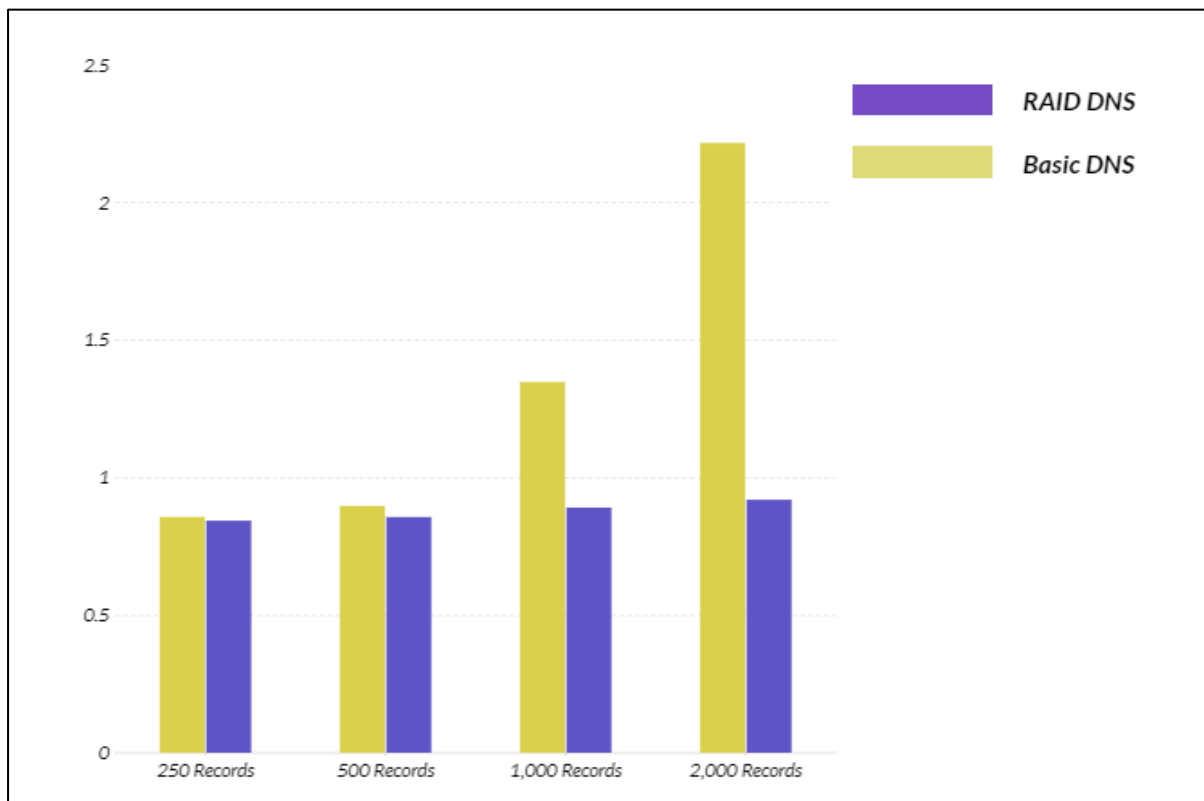


Figure 6: Total delay time of different storage size compared to a basic DNS.

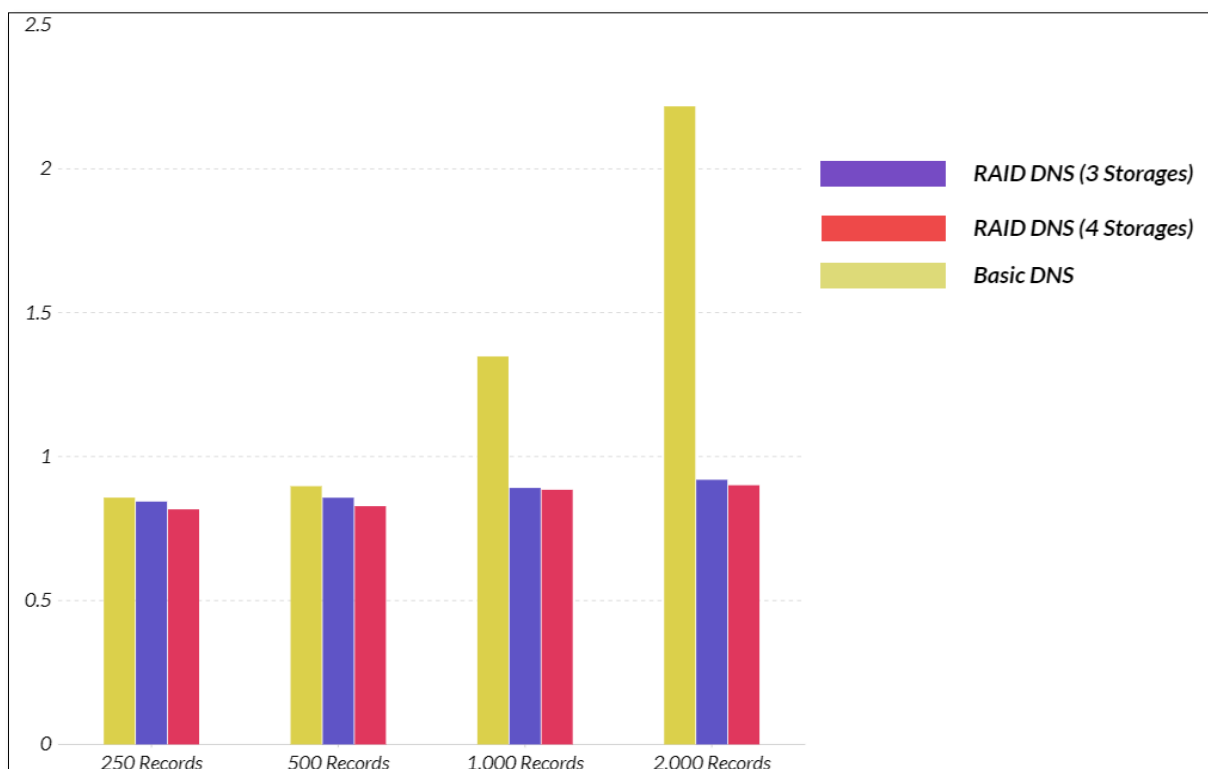
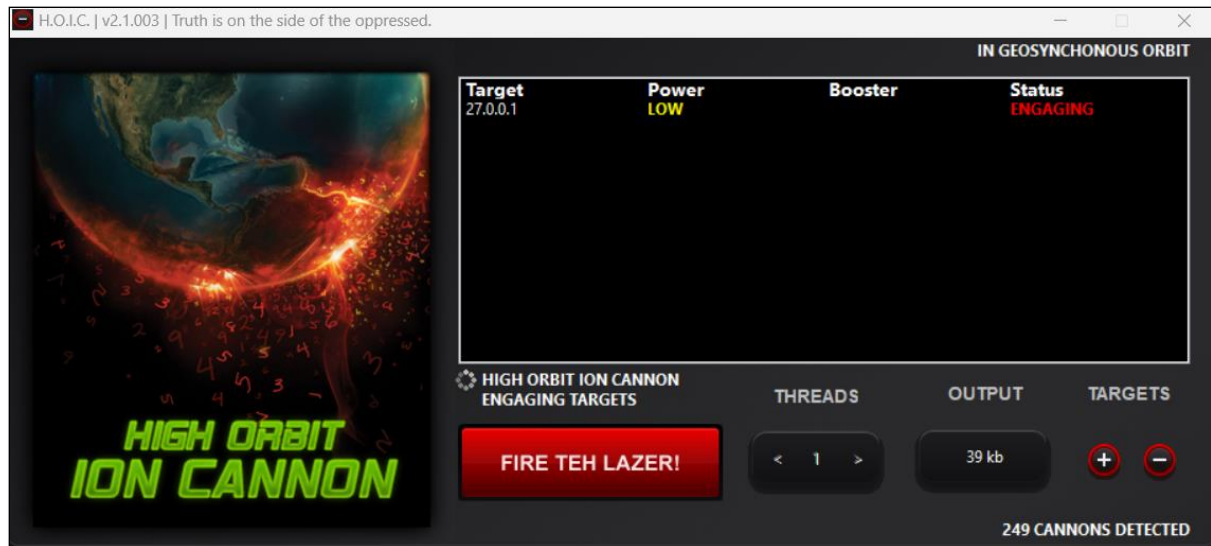


Figure 7: Total delay time of different storage size using 3 storages and 4 compared to a basic DNS.

The result was taken from an external database file that contains over 300 records, and it took our program less than one second to get the response of our request.

Now let's try performing a DoS/DDoS attack on the server and check if the attack will affect the server's performance or not.



Performing DoS attack using High Orbit Ion Cannon (HOIC)

```

In [2]: runfile('D:/jawad/AAU/AAU Y4 S2/Capstone Project II/DNS
Code/DNS_FunctionalityCode.py', wdir='D:/jawad/AAU/AAU Y4 S2/
Capstone Project II/DNS Code')
Test successful.. Requested www.AIAinUni.com, and received
176.32.108.166
Total time: 1.47626 Second

In [3]:

```

Results after requesting [www.AIAinUni.com](http://www.AIAinUni.com) during the DoS attack

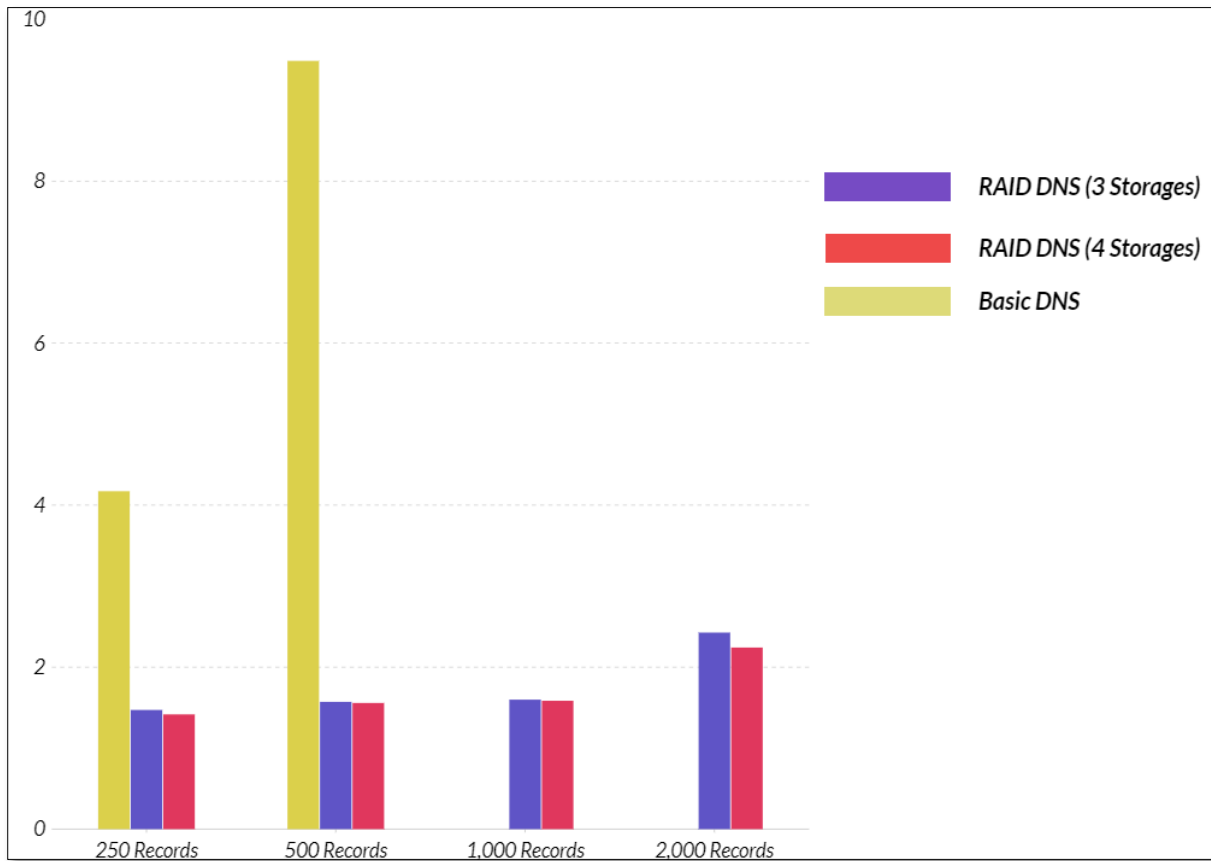


Figure 8: Total delay time of different storage size using 3 storages and 4 compared to a basic DNS during a DDoS attack using High Orbit Ion Cannon (HOIC) on one of their storages.

From the results, we can conclude that the server is experiencing increased traffic or workload as a result of the attack. However, since the network uses three DNS servers that share part of the main records, it is less likely that the server would become unresponsive due to the increased traffic. The remaining two DNS servers could still perform and retrieve the required data, without losing data or time in the process. And as we can see in figure 8, a basic DNS server delay was much higher than the RAID DNS server and furthermore the basic DNS server shut down its service on 1,000 and 2,000 records while the RAID DNS server continued to workout along with a short delay time.

For our next experiment we will be performing a 100 DNS request to our server to measure the total time taken for the response to be retrieved as well as total delay in comparison to sending a single DNS request using the following code:

```
1. import socket

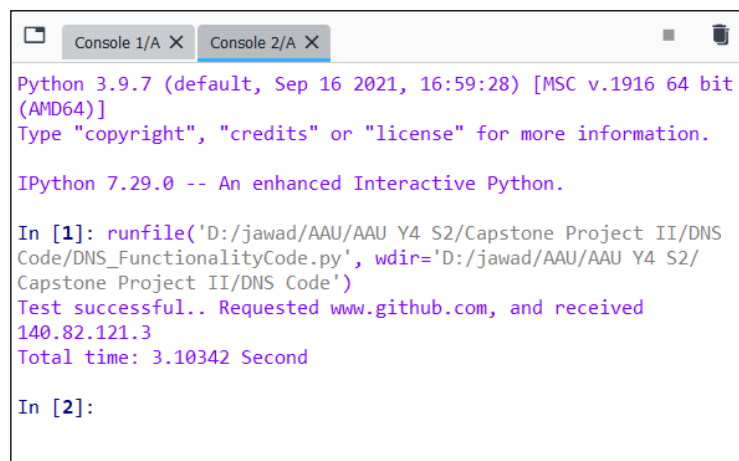
2. dns_ip = "127.0.0.1"
3. dns_port = 5000
4. num_requests = 100

5. dns_query =
    b"\x00\x01\x01\x00\x00\x01\x00\x00\x00\x00\x00\x03\x77\x77\x77\x0
    6\x67\x6f\x6f\x67\x6c\x65\x03\x63\x6f\x6d\x00\x00\x01\x00\x01"
6. sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

7. # Send DNS requests
8. for i in range(num_requests):
9.     sock.sendto(dns_query, (dns_ip, dns_port))

10. sock.close()
```

Now we try requesting different website, [www.github.com](https://www.github.com) as an example:



```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

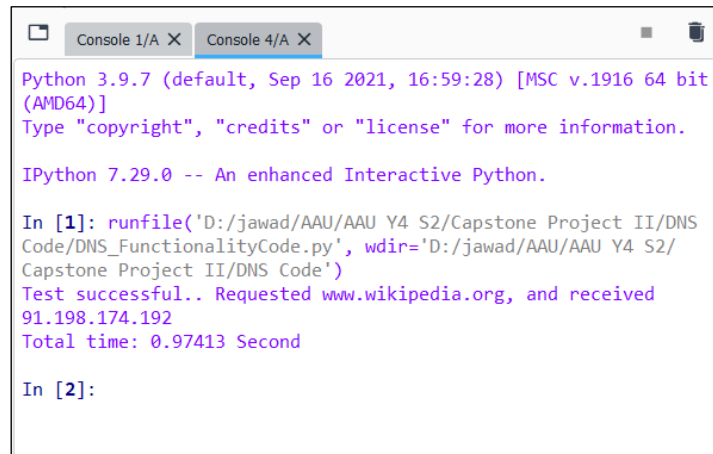
In [1]: runfile('D:/jawad/AAU/AAU Y4 S2/Capstone Project II/DNS
Code/DNS_FunctionalityCode.py', wdir='D:/jawad/AAU/AAU Y4 S2/
Capstone Project II/DNS Code')
Test successful.. Requested www.github.com, and received
140.82.121.3
Total time: 3.10342 Second

In [2]:
```

We can see that it took the server 3.10342 seconds to send the response back, which means that the server's performance is impacted by the number of requests it is handling simultaneously. As the number of requests increases, the response time of the server also increases, leading to a delay in delivering the response to the client. Which is why we mentioned load balancing as a design constraint, because it is very important to distribute the balance across all the servers instead of one main server.

In the following experiment, we added a fourth DNS server storage and tested the server's functionality and performance when dealing with a single request and more than 100 requests and we were able to obtain the following results:

Sending one single request results:



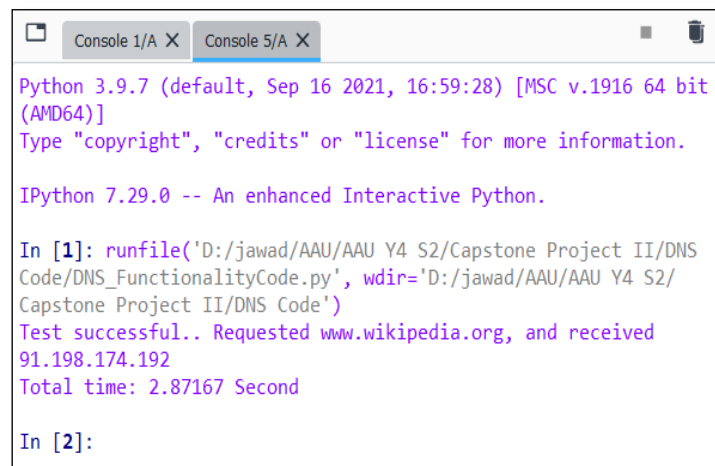
```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/jawad/AAU/AAU Y4 S2/Capstone Project II/DNS
Code/DNS_FunctionalityCode.py', wdir='D:/jawad/AAU/AAU Y4 S2/
Capstone Project II/DNS Code')
Test successful.. Requested www.wikipedia.org, and received
91.198.174.192
Total time: 0.97413 Second

In [2]:
```

Sending request with the server being overloaded:



```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/jawad/AAU/AAU Y4 S2/Capstone Project II/DNS
Code/DNS_FunctionalityCode.py', wdir='D:/jawad/AAU/AAU Y4 S2/
Capstone Project II/DNS Code')
Test successful.. Requested www.wikipedia.org, and received
91.198.174.192
Total time: 2.87167 Second

In [2]:
```

We can see that after a fourth DNS storage server was added, it potentially improved the total time and delay of the system. With an additional server, the DNS server would have more options for storing and retrieving data, which could reduce the queuing and transmission time.



Our experiments showed that the DNS server with RAID-5 methodology was able to withstand DDoS attacks and DNS flooding with minimal impact on performance. We also observed that the system's response time was slightly improved when using RAID-5 methodology, indicating that the additional redundancy did not significantly impact performance.

This observation is in line with the expected benefits of using RAID-5 methodology for DNS servers. By distributing the data across multiple disks and using parity information, RAID-5 allows for fault tolerance and data redundancy, which can help protect against DDoS attacks and DNS flooding. Additionally, the performance impact of using RAID-5 is typically minimal, as the system can still access the necessary data with only a slight increase in latency.

Our results suggest that DNS servers with RAID-5 methodology may be a promising approach to improving the resilience of DNS systems against DDoS attacks and DNS flooding. But overall, these results suggest that using RAID-5 methodology can be an effective solution for improving the reliability and performance of DNS servers, particularly in the face of DDoS attacks and DNS flooding. However, it is important to note that no system is completely immune to such attacks, and additional security measures may still be necessary to fully protect against them.

## CHAPTER 6

# Conclusion And Future Work

Our project proposes a new protection technique for Domain Name Systems (DNS) based on RAID technology, which aims to improve the reliability and performance of DNS servers in the face of DDoS attacks. By using RAID-5 methodology, our system distributes the data across multiple disks and uses parity information to ensure fault tolerance and data redundancy, while also maintaining minimal impact on performance.

Our experiments demonstrated that the DNS server with RAID-5 methodology was able to withstand DDoS attacks and DNS flooding with minimal impact on performance, and the system's response time was even slightly improved compared to traditional DNS servers. These results suggest that our proposed solution can be an effective way to improve the reliability and performance of DNS servers.

However, there are some limitations and areas for future work. For example, our experiments were conducted in a controlled environment, and it is important to test our proposed solution in a more realistic and dynamic setting to fully evaluate its effectiveness. Additionally, further research is needed to explore other potential solutions and improvements, such as using more advanced RAID configurations or incorporating machine learning techniques to better predict and prevent DDoS attacks.

Additionally, our project offers a promising new approach to addressing the longstanding problem of DDoS attacks on DNS servers, and we hope that it will inspire further research and development in this area. We summarise our project goals and achievements and reflect on the limitations and challenges encountered during the project. We also discuss potential future work and improvements to the system.

In conclusion, our project demonstrated the feasibility and effectiveness of combining DNS servers with RAID-5 methodology to improve the resilience of DNS systems against DDoS attacks and DNS flooding. Our results showed that the system was able to withstand various types of attacks while maintaining its performance.

For future work, we suggest exploring other RAID methodologies and evaluating their impact on DNS server performance. We also suggest implementing additional security measures, such as firewalls and intrusion detection systems, to further improve the system's resilience against attacks.

## References

- [1] Postel, J. (1994). *Domain name system structure and delegation* (No. rfc1591).
- [2] Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005). *DNS security Introduction and requirements* (No. rfc4033).
- [3] Eastlake 3rd, D. (1999). *Domain name system security extensions* (No. rfc2535).
- [4] Schulze, M., Gibson, G., Katz, R., & Patterson, D. A. (1989, January). How reliable is a RAID? In *COMPCON Spring 89* (pp. 118-119). IEEE Computer Society.
- [5] Chen, P. M., Lee, E. K., Gibson, G. A., Katz, R. H., & Patterson, D. A. (1994). RAID: High-performance, reliable secondary storage. *ACM Computing Surveys (CSUR)*, 26(2), 145-185.
- [6] Chen, S., & Towsley, D. (1996). A performance evaluation of RAID architectures. *IEEE Transactions on computers*, 45(10), 1116-1130.
- [7] <https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020>
- [8] Kolkman, O., & Gieben, R. (2006). *DNSSEC operational practices* (No. rfc4641).
- [9] Ateniese, G., & Mangard, S. (2001, November). A new approach to DNS security (DNSSEC). In *Proceedings of the 8th ACM conference on Computer and Communications Security* (pp. 86-95).
- [10] Ariyapperuma, S., & Mitchell, C. J. (2007, April). Security vulnerabilities in DNS and DNSSEC. In *The Second International Conference on Availability, Reliability and Security (ARES'07)* (pp. 335-342). IEEE.
- [11] Herzberg, A., & Shulman, H. (2013, October). DNSSEC: Security and availability challenges. In *2013 IEEE Conference on Communications and Network Security (CNS)* (pp. 365-366). IEEE.
- [12] of Hoang, N. P., Polychronakis, M., & Gill, P. (2022, March). Measuring the Accessibility Domain Name Encryption and Its Impact on Internet Filtering. In *International Conference on Passive and Active Network Measurement* (pp. 518-536). Springer, Cham.

- [13] Cheng, Y., Chai, T., Zhang, Z., Lu, K., & Du, Y. (2022). Detecting malicious domain names with abnormal whois records using feature-based rules. *The Computer Journal*, 65(9), 2262-2275.
- [14] Mockapetris, P., & Dunlap, K. J. (1988, August). Development of the domain name system. In *Symposium proceedings on Communications architectures and protocols* (pp. 123-133).
- [15] Atkins, D., & Austein, R. (2004). *Threat analysis of the domain name system (DNS)* (No. rfc3833).
- [16] Mockapetris, P. V. (1987). RFC1034: Domain names-concepts and facilities.
- [17] Mockapetris, P. V. (1987). Rfc1035: Domain names-implementation and specification.
- [18] Mockapetris, P. (1983). *Domain names: Implementation specification* (No. rfc883).
- [19] Persson, J., & Evertsson, G. (2002). RAID Systems. *Blekinge Institute of Technology, Sweden, Research Paper*, 1-10.
- [20] Ljungquist, A., Claesson, J., & Bondesson, T. (2003). RAID Technology.
- [21] Sanders, D. A., Cremaldi, L. M., Eschenburg, V., Godang, R., Lawrence, C. N., Riley, C., ... & Petravick, D. L. (2003). Terabyte IDE RAID-5 Disk Arrays. *arXiv preprint physics/0306037*.
- [22] Rahman, P. A., & Shavier, G. D. K. N. F. (2018, March). Reliability model of disk arrays RAID-5 with data striping. In *IOP Conference Series: Materials Science and Engineering* (Vol. 327, No. 2, p. 022087). IOP Publishing.
- [23] Kuratti, A., & Sanders, W. H. (1995, April). Performance analysis of the RAID 5 disk array. In *Proceedings of 1995 IEEE International Computer Performance and Dependability Symposium* (pp. 236-245). IEEE.
- [24] Chen, S. Z., & Towsley, D. (1993). The design and evaluation of RAID 5 and parity striping disk array architectures. *Journal of Parallel and Distributed Computing*, 17(1-2), 58-74.
- [25] Mishra, S. K., & Mohapatra, P. (1996, August). Performance study of RAID-5 disk arrays with data and parity cache. In *Proceedings of the 1996 ICCP Workshop on Challenges for Parallel Processing* (Vol. 1, pp. 222-229). IEEE.
- [26] Menon, J. (1995, August). A performance comparison of RAID-5 and log-structured arrays. In *Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing* (pp. 167-178). IEEE.