

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018, Karnataka, INDIA



PROJECT REPORT

on

“An Efficient Cloud Based Approach for Decentralized DNS System”

Submitted in partial fulfillment of the requirements for the VIII Semester

Bachelor of Engineering IN COMPUTER SCIENCE AND ENGINEERING

For the Academic year

2016-2017

BY

Arisha Siddiqui	1PE13CS032
Shashish Jha	1PE13CS139
Sindhoor Tilak S Hegde	1PE13CS148
Venugopala B V	1PE14CS431

Under the Guidance of

Dr. Annapurna D.

HOD, Dept. of ISE

PESIT-BSC, Bengaluru-560100



Department of Computer Science and Engineering

PESIT BANGALORE SOUTH CAMPUS

Hosur Road, Bengaluru -560100

PESIT BANGALORE SOUTH CAMPUS

Hosur Road, Bangalore -560100

Department of Computer Science and Engineering



CERTIFICATE

Certified that the project work entitled “An Efficient Cloud Based Approach for Decentralized DNS System” is a bonafide work carried out by Arisha Siddiqui, Shashish Jha, Sindhoor Tilak S Hegde and Venugopala B.V bearing USN 1PE13CS032, 1PE13CS139, 1PE13CS148 and 1PE13CS431 respectively, students of PESIT Bangalore South Campus in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2016-2017. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated and the project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signatures:

Project Guide
Dr. Annapurna D
HOD, Dept. of ISE
PESIT-BSC, Bengaluru

Head Dept of CSE
Mr. Sandesh B J
Associate Prof., Dept. of CSE,
PESIT-BSC, Bengaluru

Director/Principal
Dr. SuryaPrasad J
PESIT-BSC, Bengaluru

External Viva

Name of the Examiners

1. _____
2. _____

Signature with date

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned my effort with success.

We are indebted to our Guide, **Dr. Annapurna D**, HOD, Department of Information Science and Engineering, PESIT-Bangalore South Campus, who has not only coordinated our work but also given suggestions from time to time.

We are also extremely grateful to our Project Co-ordinators, **Dr. Snehanshu Saha**, Professor, **Mrs. Pooja Agarwal**, Associate Professor, **Mrs. Vandana M L** and **Mrs. Jyoti Desai**, Assistant Professors, Department of Computer Science and Engineering, PESIT Bangalore South Campus, for their constant support and advice throughout the course of preparation of this document.

We are greatly thankful to **Mr. Sandesh B J**, Associate Professor and HOD, Department of Computer Science and Engineering, PESIT Bangalore South Campus, for his able guidance, regular source of encouragement and assistance throughout this project.

We would like to express our immense gratitude to **Dr. Suryaprasad J**, Director and Principal, PESIT Bangalore South Campus, for providing us with excellent infrastructure to complete our project work.

We gratefully acknowledge the help lent out to us by all faculty members of the Department of Computer Science and Engineering, PESIT Bangalore South Campus, at all difficult times. We would also take this opportunity to thank our college management for the facilities provided during the course of the project. Furthermore, we acknowledge the support and feedback of my parents and friends.

Arisha Siddiqui
Shashish Jha
Sindhoor Tilak S Hegde
Venugopala BV

ABSTRACT

DNS or Domain Name Server, is a service provided to the user for the translation of domain names to IP addresses. This service plays a critical role in working of the Internet. Without a DNS Server, whole of the internet comes to a standstill. DNS is vital and critical to the Internet infrastructure. Over the years, numerous flaws and loopholes have been uncovered in the existing model of DNS. The whole structure of DNS relies on a trust model. The system is not transparent and under tight control of very few organizations. We need a system that forces transparency as well decentralize the whole system. Also, we need to solve the problems of Domain Record manipulation, false DNS resolution. The system must be resistant to all these sort of problems. We make use of blockchain technology, to overcome these problems.

The goal of our project is to develop a system that is totally decentralized and also develop an algorithm that optimizes the DNS service.

CONTENTS

Acknowledgement	i
Abstract	ii
Contents	iii
List of figures	vi
List of Tables	vii

Chapter 1

1.	Introduction	1
1.1	Purpose of the project	1
1.2	Scope	1
1.3	Definitions, acronyms and abbreviations	2
1.4	Existing System	3
1.5	Proposed system	3
1.5.1	A Public Ledger	4
1.5.2	Blockchain	5
1.5.3	Authentication	5
1.5.4	Trusted Entities	5
1.6	Statement of the Problem	6
1.7	Summary	7

Chapter 2

2.	Software Requirements Specifications	8
2.1	Operating Environment	8
2.1.1	Hardware Requirements	8
2.1.2	Software Requirements	9
2.2	Functional Requirements	9
2.3	Non functional requirements	10
2.4	User characteristics	10
2.5	Applications of System	10
2.6	Advantages of System	11

2.7	Summary	11
Chapter 3		
3.	High Level Design	12
3.1	Design Considerations	12
3.2	System Architecture	13
3.3	Data Flow Diagram	14
3.3.1	Algorithm	15
3.4	Sequence diagram	16
3.5	Summary	17
Chapter 4		
4.	Detailed design	18
4.1	Purpose	18
4.2	User Interface Design	18
4.2.1	Software Graphical User Interfaces	19
4.3	Module 1: BlockChain	19
4.4	Module 2 : Cloud Simulation	21
4.5	Module 3: Algorithm	22
4.6	Summary	23
Chapter 5		
5.	Implementation	24
5.1	Programming Language Selection	24
5.2	Coding Conventions	25
5.2.1	Naming Convention	25
5.2.2	Function Prototypes and Methods	26
5.3	Platform Selection	26
5.4	Graphical User Interface Design	27
5.4.1	Cloud Analyst	27
5.5	Summary	28
Chapter 6		
6.	Testing	29
6.1	Software Testing	29
6.2	Testing Process	30

6.2.1	Levels of Testing	30
6.3	Unit testing	31
6.3.1	Unit Testing of Blockchain Module	31
6.3.2	Unit Testing of Cloud Sim Module	32
6.4	Integration Testing of Modules	33
6.5	Summary	33
Chapter 7		
7.	Experimental Results	34
7.1	Blockchain	34
7.2	Cloud Simulation Results	36
7.3	Summary	38
Chapter 8		
8.	Conclusion	39
8.1	Limitations of the Project	40
8.2	Future Enhancement	40
References		41

LIST OF FIGURES

1.1	Block Chain	4
3.1	System Architecture	13
3.2	Algorithm Flow Diagram	14
3.3	Sequence Diagram	16
4.1	Blockchain Interface	20
4.2	FrontEnd	20
4.3	Cloud Analyst GUI	21
4.4	Algorithm Output	22
7.1	Blockchain Results	34
7.2	Output of DNS Records stored	35
7.3	Offset of Records	35
7.4	Hourly Response Time Graph	36
7.5	Loading Times	36
7.6	User Base Mean Response Times	37
7.7	Data Center Service Times	38

LIST OF TABLES

5.1	Project Files	25
5.2	Function Prototypes	26
6.1	Unit Test Case 1	31
6.2	Unit Test Case 2	32
6.3	Unit Test Case 3	32
6.4	Unit Test Case 4	32
6.5	Integration Test Case 1	33

CHAPTER 1

INTRODUCTION

The Domain Name System (DNS) is a technology standard for managing names of public Web sites and other Internet domains. DNS technology allows you to type names into your Web browser like abc.com and your computer to automatically find that address on the Internet. DNS servers are spread around the world.

A DNS server is any computer registered to join the Domain Name System. A DNS server runs special-purpose networking software, features a public IP address, and contains a database of network names and addresses for other Internet hosts.

The DNS is a distributed database system. Only the 13 root servers contain the complete database of names and addresses. All other DNS servers are installed at lower levels of the hierarchy and maintain only certain pieces of the overall database.

1.1 Purpose

The existing legacy DNS (Domain Name Service) systems on the internet, have issues ranging from slowness, does not support updates, vulnerable to DoS (Denial of Service). The existing system is filled with loopholes and is not robust. There is a need to change the existing architecture to be more decentralized and transparent. The whole DNS works on the trust model of the TLD.

1.2 Scope

The scope of this system is very critical. The DNS plays a critical role in supporting the Internet infrastructure by providing a distributed and fairly robust mechanism that

resolves Internet host names into IP addresses and IP addresses back into host names. The DNS also supports other Internet directory-like lookup capabilities to retrieve information pertaining to DNS Name Servers, Canonical Names, Mail Exchangers, etc. Unfortunately many security weaknesses surround IP and the protocols carried by IP. The DNS is not immune to these security weaknesses. The accuracy of the information contained within the DNS is vital to many aspects of IP based communications.

This system makes this existing system more robust, without affecting the operation of the existing Internet.

1.3 Definitions, Acronyms and Abbreviation

JAVA: Java is an object-oriented computer programming language. It is concurrent, class based and object-oriented in nature. It is designed for the use in the distributed environment of the internet. It has the look and feel of C++.

JRE: It stands for Java Runtime Environment. It is a part of JDK. JRE is used as an interpreter of byte code to machine code.

JVM: It stands for Java Virtual Machine. JVM is the runtime engine of the Java Platform, which allows any program written in Java or other language compiled into Java bytecode to run on any computer that has a native JVM.

JAR: It stands for Java Archive. It is a package file format typically used to aggregate many java class files and associated metadata and resources (text, images etc.) into one file to distribute application software or libraries on the Java platform.

DataCenter: Datacenter class is a Cloud Resource whose host lists are virtualized. It deals with processing of VM queries (i.e., handling of VMs) instead of processing Cloudlet-related queries.

PKI: It stands for Public Key Infrastructure. A public key infrastructure is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.

Blockchain: is a distributed database that maintains a continuously growing list of ordered records called *blocks*. Each block contains a timestamp and a link to a previous block.

TLD: TLD stands for Top Level Domain. It controls the domain names like (.com, .org etc.)

CA: CA stands for Certificate Authority. Certificate Authority is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate.

1.4 Existing System

Currently, the present model of DNS works under a hierarchical structure. Each root server contains a reference to the TLD. This TLD then references it to a Authoritative Server to translate the IP address. The DNS relies on the trust of TLD and there is no existing mechanism for verification if the records are altered or not. The whole DNS architecture relies on the trust of these servers. We need a robust system that eliminates this problem and gives rise to a open transparent system.

1.5 Proposed System

Since we want a system which can coexist with the existing infrastructure, this system provides the following operations:

- Registrations

- Key updates (of the public key associated with the domain)
- Revocations

1.5.1 A Public Ledger

In this system, every TLD has a public ledger, maintained by TLD.

With every update (any of the above actions) an entry is appended to this ledger. The ledger is broken up into blocks. A block contains all updates in the last 30 minutes.

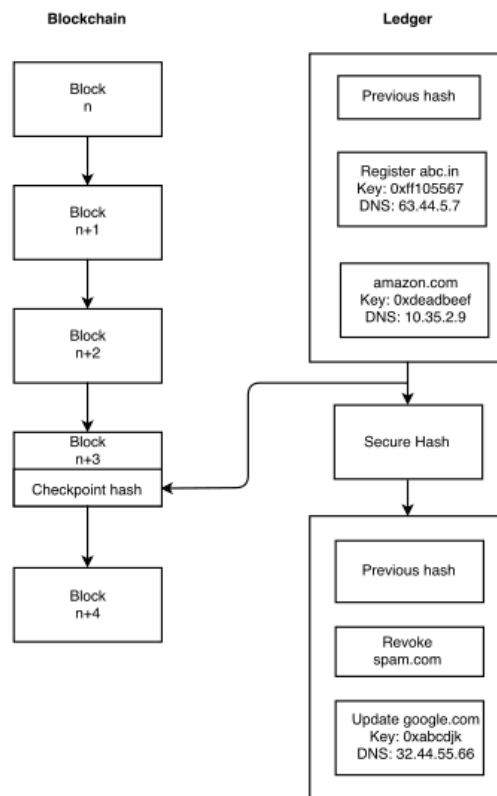


Fig 1.1 Block Chain

1.5.2 Blockchain

Rather than stuffing these into a blockchain directly, the latest additions to the ledger are hashed, together with the hash of the last update block. Thus creating a chain of update blocks.

This hash is added to a public blockchain (e.g the Bitcoin blockchain). Information for locating the hash is served by TLD together with the ledger (this information could also be included in the following block). TLD must insert a new checkpoint in the blockchain every 30 min and there is one ledger block for every checkpoint.

1.5.3 Authentication

Public keys are added to the DNS records stored in the public ledger. These keys are not added to the legacy DNS entries, but may also be validated against the existing PKI.

When Server Admin generates a new key for alice.com, she requests that TLD updates the ledger to reflect this. In addition she contacts a CA (Certificate Authority) to have the key signed and thus maintains backwards compatibility.

The user keeps a copy of the blockchain and the public ledger (downloaded from TLD). When visiting a site user queries the local copy of the public ledger (verified against the blockchain) and finds the corresponding public key.

1.5.4 Trusted Entities

The Trusted Entities play a major role in this approach. The user needs to follow the blockchain. If he goes offline, he needs to download the entire blockchain and TLD ledger before visiting any site.

The user needs to store a large amount of data. The ledger is potentially large and he needs to store one for every TLD he wishes to use (potentially hundreds).

This can be solved by offloading the work to a trusted entity (Trusted Entity). The user may have multiple trusted entities. He may switch them at any time (or operate his own) and they are not globally trusted by all users.

He may choose to:

- Cross check these against each other (since they should all return the same key)
- Have one for each TLD (an entity may only follow a subset of TLDs)
- Rotate these for privacy reasons.

All the Trusted Entities are cloud services owned by Private or Public organizations. The below algorithm figures the most optimal Cloud Service enabling faster accessing.

The list of Trusted Entities are listed in a tree structure. We employ the Alpha-beta Pruning which is adversarial graph search algorithm to identify the nearest and best trusted entity for the user.

1.6 Statement of the Problem

In this project, we have developed an newer architecture of the working of DNS. This new approach results in a decentralized and more transparent working of the DNS. In addition to it, we've developed an algorithm to improve the resolution time of DNS queries. This model is built in the following phases:

- Designing the blockchain architecture
- Simulation of Cloud Trusted Entities
- Development of Algorithm and integrating with Cloud Simulation

1.7 Summary

This chapter describes, in brief the idea of the project. It talks about the intent of the project, the definitions of various terminologies used in the document, abbreviations and scope of the project.

CHAPTER 2

SYSTEM REQUIREMENT SPECIFICATION

2. Software requirement specification

Requirement specification gives a description of the intended environment for the development of the system under consideration. It is the activity of translating the information collected during survey and analysis into a requirement document. Requirements Analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Systematic requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering , requirements capture, or requirements specification.

The term requirements analysis can also be applied specifically to the analysis proper (as opposed to elicitation or documentation of the requirements, for instance).Requirements analysis is critical to the success of a development project. Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

2.1 Operating Environment

The system is designed to run on Windows or Linux Operating System.

2.1.1 Hardware Requirements

A Computer with:

- Processor: Any Intel or AMD x86 processor supporting SSE2 instruction set.
- Disk space: Minimum of 2 GB

- RAM: 512 MB

2.1.2 Software Requirements

- **Operating System:** Windows XP and above or Ubuntu 12.04 or above.
- **Programming language:** Python 2.7, PHP 5.6 or above for the design and implementation of the application and Java.
- **Packages:** Java Run Time installed along with CloudSim and CloudAnalyst.

2.2 Functional Requirements

These are the statement of services the system should provide and how the system should react for particular inputs and how the system should behave in the particular situations. They are: A function is described as a set of inputs, the behavior and outputs. Functional requirements may be calculations, technical details, data manipulation, processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation.

The functional requirements of the system are:

- The control of the system must lie with the TLD organizations.
- The TLD must verify the ownership of the domain names before registering.
- The Trusted Entity lists are maintained separately and periodically reviewed.

2.3 Non-Functional Requirements

Non-functional requirements are requirements which impose constraints on the design or implementation such as performance engineering requirements, quality standards, or design constraints.

The non-functional requirements are as follows:

- **Reliability:** is essential for all cloud systems- in order to support today's data center type applications in a cloud, reliability is considered one of the main features to exploit cloud capabilities. Reliability denotes the capability to ensure constant operation of the system without disruption, i.e. no loss of data, no code reset during execution etc. Reliability is typically achieved through redundant resource utilization.
- **Security:** Care must be taken against unauthorized access to the TLD ledger.
- **Scalability:** The system must be carefully considered while it scales up.
- **Availability:** The measure of time that the system is up and running correctly, this service is available 24*7 since it is hosted on the Internet.

2.4 User Requirements

User characteristics define the different users who can use the application and benefit from it. User characteristics will help us in understanding the target customers and give us a fair idea as for whom we need to develop this system.

The users perform the DNS resolution which is served by this system.

2.5 Applications of the System

This system can serve as an alternative or co-exist with the current DNS system. This whole model of DNS is used to optimize the disadvantages of the existing system. It can be

implemented with existing cloud datacenters. This implementation also helps in optimizing the response time and service.

2.6 Advantages of the System

This application has the following advantages:

- TLD may implement this system if he pleases (or opt-out)
- TLD does not need to contact ICANN (Internet Corporation for Assigned Names and Numbers) to implement this.
- TLD still retains full control of the TLD (including revocations)

2.7 Summary

This chapter talks about requirement specification of the software. It focuses on the hardware and software requirements , also the environment needed to run the system. It also describes the users that interact with this system . It gives an outline of what is being needed to develop , design and test the application.

CHAPTER 3

HIGH LEVEL DESIGN

Design is concerned with identifying software components specifying relationships among components, specifying software structure and providing blue print for the document phase. A software design is description of the structure of the software to be implemented, the data which is a part of the system, the interfaces between the system components and sometimes the algorithm used. Designers do not arrive at a finished design immediately but develop the design iteratively through a number of different versions. The design process involves adding formality and detail as the design is developed with constant backtracking to correct earlier designs.

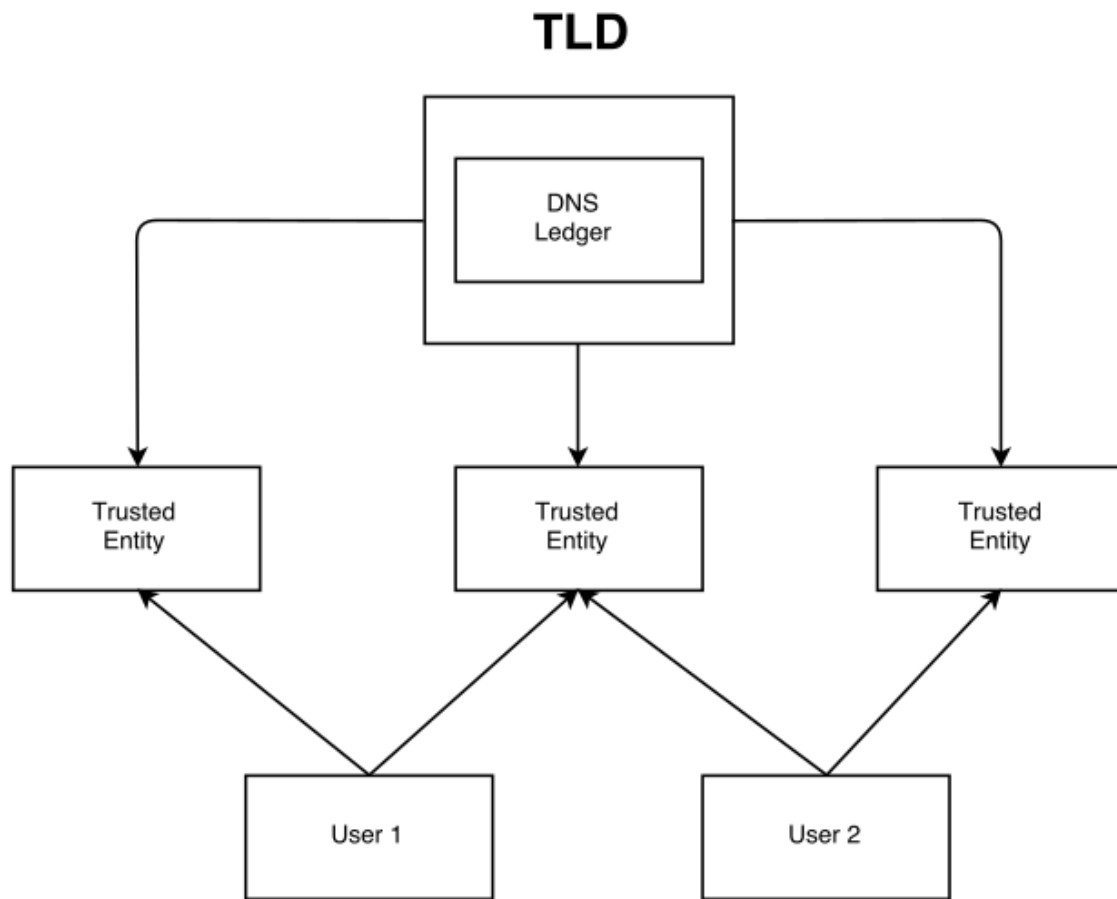
3.1 Design Considerations

This section mainly covers the design technique of the entire system which involves the implementation of three modules which are as follows:

- Implementation of the Blockchain ledger
- Simulation of the Cloud-Based Trusted Entities
- Implementation of the algorithm to choose the best data center.

3.2 System Architecture

The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces.

**Fig 3.1 System Architecture**

The system contains a blockchain based ledger which is maintained by TLD organization. Different Trusted Entities around the world make copies of this blockchain ledgers. The users can then use the any of the trusted entities for the DNS resolution of the sites.

3.3 Data Flow Diagram

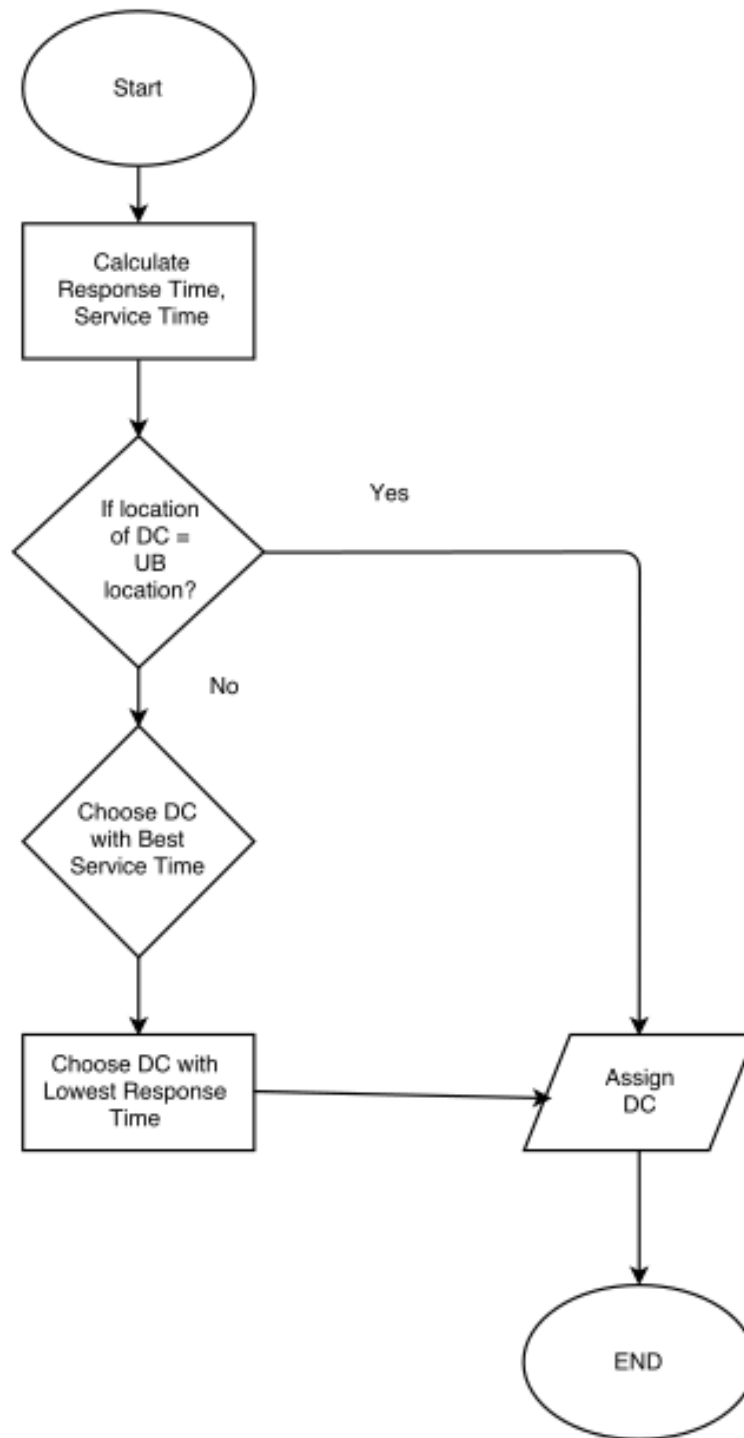


Fig 3.2 Algorithm Flow Diagram

3.3.1 Algorithm

- Calculate the Response Time and Service Times for each data centers
- Choose a datacenter from the list of datacenters
- Check if the location of the datacenter is same as the User Base location
- If true, assign the datacenter to the userbase
- Else, assign the datacenter with the lowest response time and service time.

3.4 Sequence Diagram

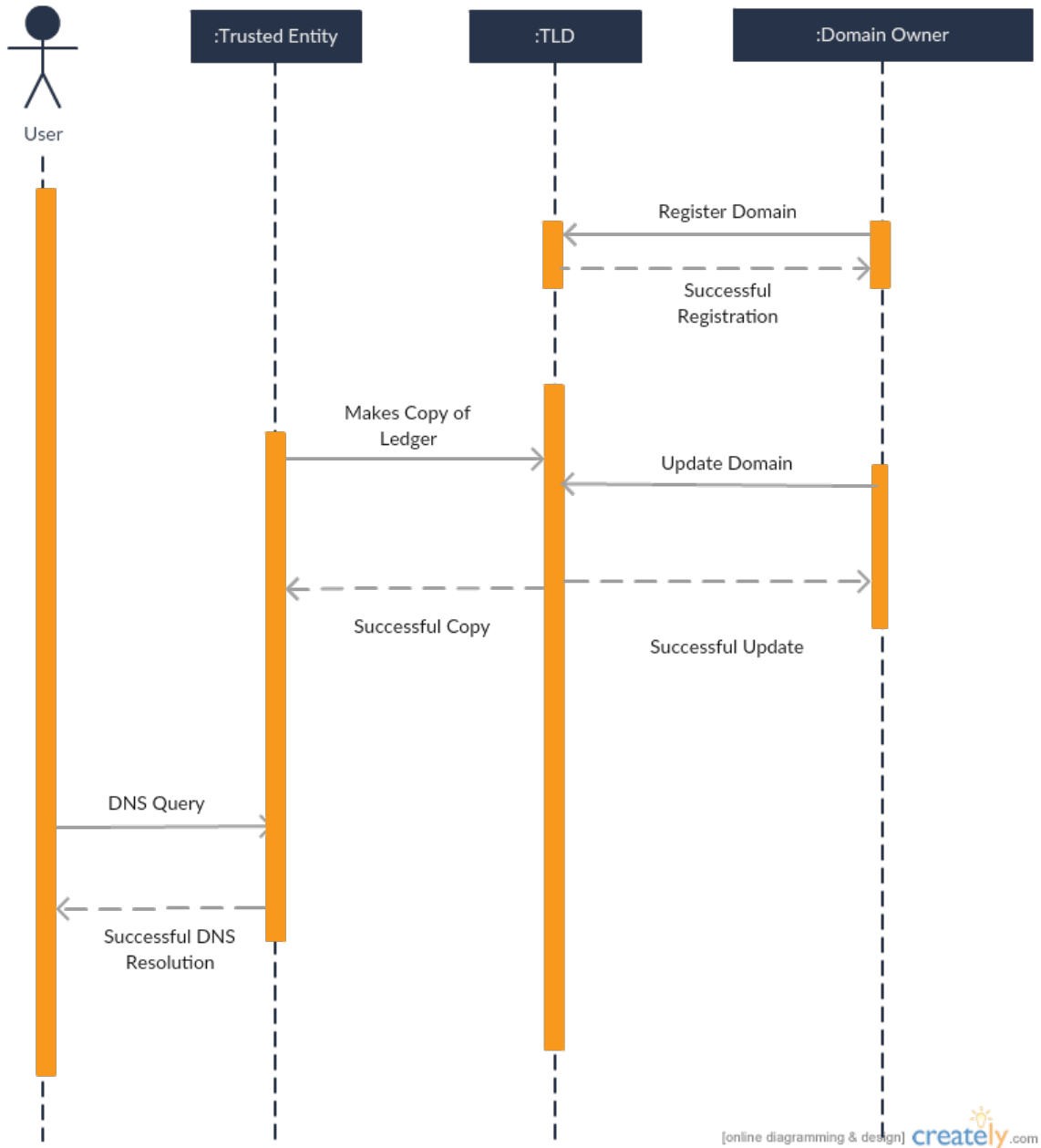


Fig 3.3 Sequence Diagram of DNS Resolution

3.5 Summary

In this chapter we discussed the overview of the entire system. We identified various modules and the functionalities. We understood the constraints and flow of the system. The flow of each of the modules is discussed here. The sequence diagram shows the interaction of the user with the system.

CHAPTER 4

DETAILED DESIGN

4.1 Purpose

Detailed design is such a fundamental necessity to manufacturers that it exists at the intersection of many product development processes. As a core engineering process, detailed design transforms concept alternatives, preliminary physical architectures, design specifications and refined and all accompanying documentation required for manufacturing is completed in order for timely delivery to the customer of a fully defined, complete product.

4.2 User Interface Design

User interface design interfaces for machines and (UI) or user software, interface such engineering is the design of user as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals which is often called as user-centered design.

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements to create a system that is not only operational but also usable and adaptable to changing user needs.

4.2.1 Software Graphical User Interfaces

The GUI of the system is implemented in PHP and Java. The PHP generates the necessary necessary HTML code after taking the user input. This Interface is used to take input from the user to fetch the required records.

The interface of the Cloud Simulation utilizes a tool called Cloud Analyst. This tools lets you visualize the simulation by plotting it in a map. It represents the Data Centers and the User Bases.

4.3 Module 1: BlockChain

This module is the primary module that implements the DNS. BlockChain is a datastructure that is implemented to hold the records of the DNS. This data structure has a unique property to withstand against overwriting and editing. Once a record is entered, it is impossible to erase a single record with deleting the whole database. This special functionality helps to avoid manipulation of the data. When a record is entered, this data is securely encrypted using the SHA-256 algorithm using a randomly generated magic number. This encrypted text is then packed as a binary and stored in the file. To retrieve the records, the program first safely unpacks the packed binary and decrypts it using the SHA-256 key used before to decrypt it. The decrypted text is then displayed to the user.

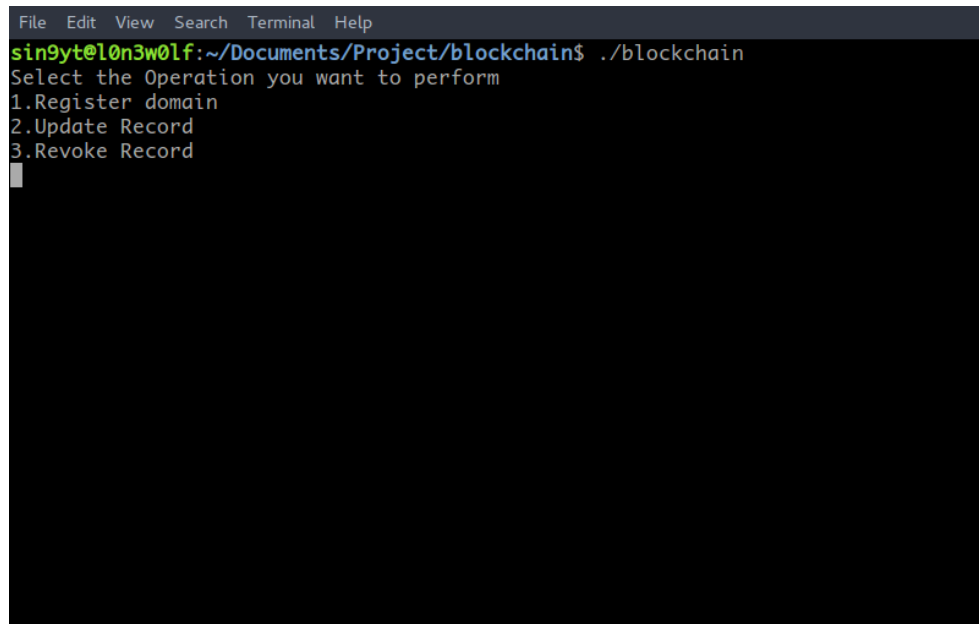


Fig 4.1 BlockChain Interface

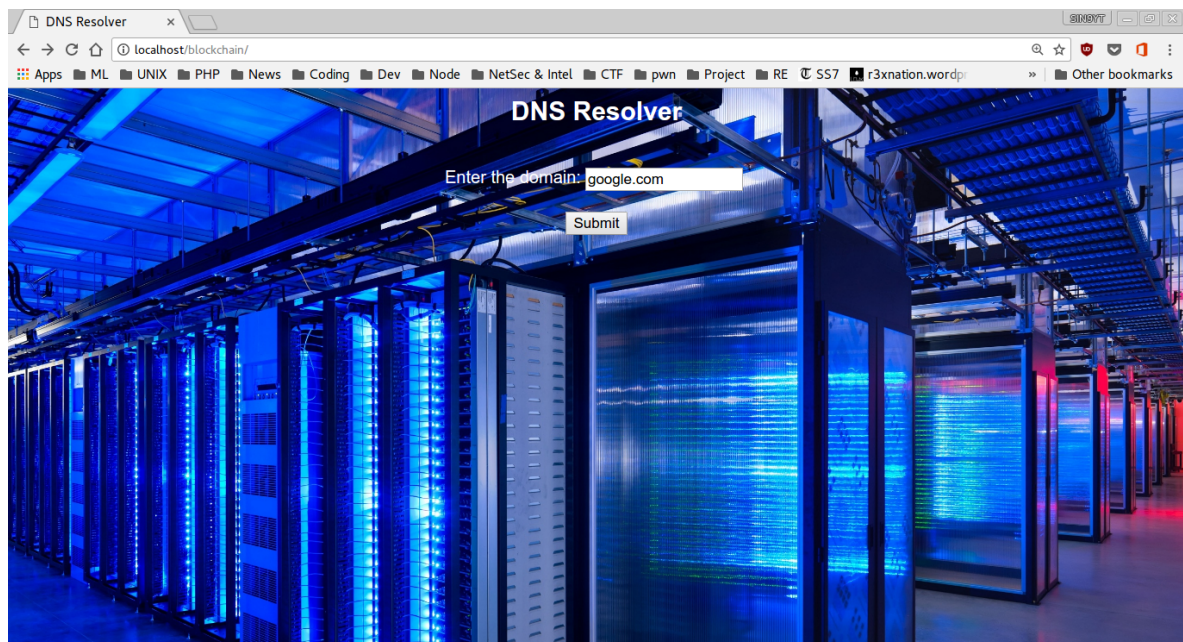


Fig 4.2 Front End

4.4 Module 2: Cloud Simulation

The simulation of the Trusted Entities is performed using a tool called CloudAnalyst. This tool is based on CloudSim which is a popular cloud simulation software used around the world. A simulation file is given as the input to the CloudAnalyst GUI which maps it to the GUI. The GUI contains a world map representing the various User Bases as well as the Data Centers. It also helps to run the simulation for over a period of time. The simulation results displays different parameters and times. It also generates few graphs of the times generated after the simulation.

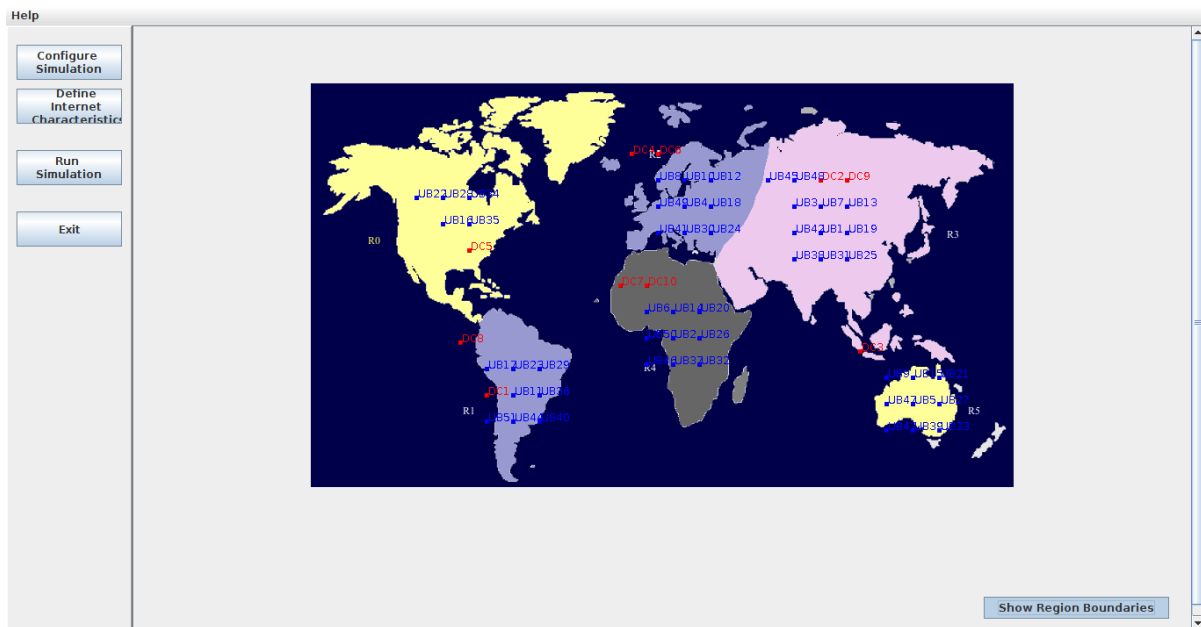
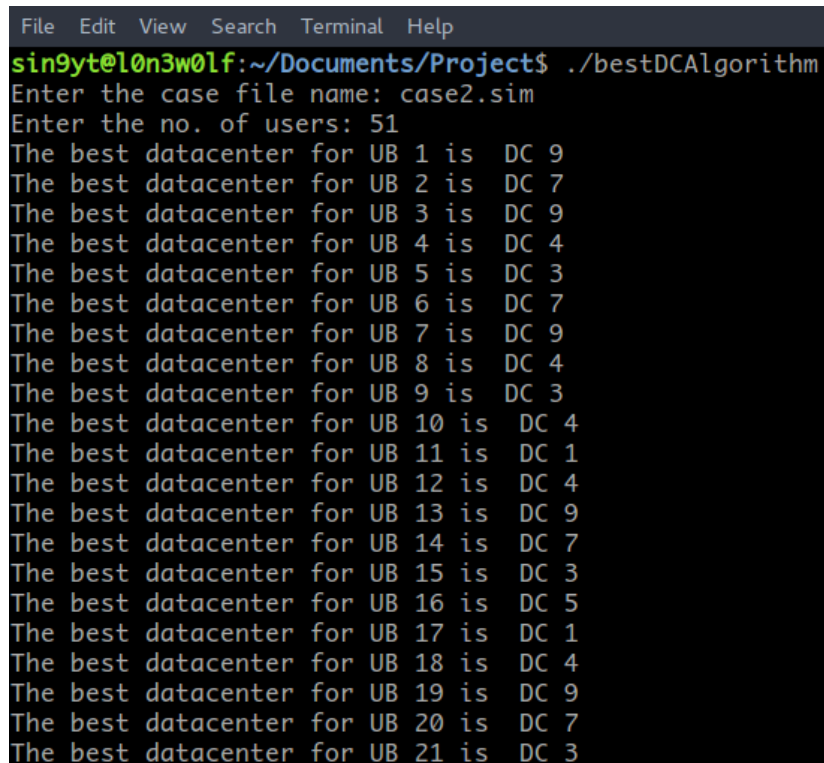


Fig 4.3 Cloud Analyst GUI

4.5 Module 3: Algorithm

This module implements the functionality of determination of the best datacenter for each User Base. The algorithm takes the results from the simulation performed before. Taking the input from the simulation results, the algorithm is applied as shown in Chapter 3.

The result is displayed by showing the name of best datacenter associated with each User Base. Along with the results, different graphs are generated based on the results showing the comparison of the times with the data centers and the user bases.



```
File Edit View Search Terminal Help
sin9yt@lon3w0lf:~/Documents/Project$ ./bestDCAAlgorithm
Enter the case file name: case2.sim
Enter the no. of users: 51
The best datacenter for UB 1 is DC 9
The best datacenter for UB 2 is DC 7
The best datacenter for UB 3 is DC 9
The best datacenter for UB 4 is DC 4
The best datacenter for UB 5 is DC 3
The best datacenter for UB 6 is DC 7
The best datacenter for UB 7 is DC 9
The best datacenter for UB 8 is DC 4
The best datacenter for UB 9 is DC 3
The best datacenter for UB 10 is DC 4
The best datacenter for UB 11 is DC 1
The best datacenter for UB 12 is DC 4
The best datacenter for UB 13 is DC 9
The best datacenter for UB 14 is DC 7
The best datacenter for UB 15 is DC 3
The best datacenter for UB 16 is DC 5
The best datacenter for UB 17 is DC 1
The best datacenter for UB 18 is DC 4
The best datacenter for UB 19 is DC 9
The best datacenter for UB 20 is DC 7
The best datacenter for UB 21 is DC 3
```

Fig 4.4 Algorithm Output

4.6 Summary

In this chapter of the report we discussed the GUI of the application, hardware and software interfaces. It also describes various modules involved in the project, which specifies how the system works.

CHAPTER 5

IMPLEMENTATION

Implementation is the realization of an application, or execution of a plan, idea, model design, specification, standard, algorithm, or policy. In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component or other computer system through programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations World Wide Web consortium recommended specification, and software tools contain implementation of programming languages.

5.1 Programming Language Selection

Python is very simple and minimalistic language. The syntax of this language is very similar to reading English. It is an interpreted language, wherein the code is read line by line , and executed. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc, etc. This also makes your Python programs much more portable. It has huge standard library helping to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), Tk, and other system-dependent stuff.

We also made use of Java. Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers “write once, run anywhere”, meaning that code that runs in one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file)

that can run on any Java Virtual Machine (JVM) regardless of computer architecture. CloudSim has its underlying platform built on Java. For enabling an easy communication between the front end and the back end, the user interface is designed using the Java Swings. The actions are performed by the Applets. The modifications are deployed over existing CloudSim library. A JAR file for the same is created.

5.2 Coding Conventions

Code conventions are a set of guidelines for a specific application that recommend programming style practices and methods for the application. These conventions usually cover the file organization, indentations, comments, declarations, statements, whitespace, naming conventions, programming practices, programming principles, and programming rules of thumb.

5.2.1 Naming Convention

File Name	Purpose
blockchain	This files generates the interface to insert, update and delete the records. It also performs the operations of writing into the encrypted binary file.
Walkchain	This file is used to display all the records stored in the binary file.
Dumpindex	This file displays the offset of the records stored in the database file.
Simulation (.sim)	This file contains all the details of the datacentres and the user bases needed for simulation.
BestDCAalgorithm	This python script implements the algorithm

Table 5.1 Project Files

5.2.2 Function Prototypes and Methods

Functions	Purpose
addblock	This function checks if there is a genesis block and adds a new block to blockchain.
write_block	This method writes the block contents to the file
update_index	This function calculates the length and offset count of the block
unpack32	This method unpacks the packed binary data
plotGraph	This function uses matplotlib to plot and draw the graphs.

Table 5.2 Function Prototypes

5.3 Platform Selection

Python is an interpreted language. As a result, a binary of python is sufficient to run the program. It is available across all the platforms i.e Mac , Windows and Linux.

Java is a part of software products developed by Oracle Corporation that helps developing cross-platform products. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. Writing in Java programming language is the primary way to produce code that will be deployed as Java byte code. The application built using Java can be run in any platform. The jar file created in this project can be deployed on any system and is tested on Windows XP and higher versions and Linux (Ubuntu) platform.

5.4 Graphical User Interface

Graphical user interface is a type of user interface that allows users to interact with electronic devices using images rather than text commands, A GUI represents the information & action available to a user through graphical icons & visual indicators such as secondary notation, as opposed to text-based interfaces, type command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements. A GUI uses a combination of technologies & devices to provide a platform that the user can interact with, for the tasks of gathering & producing information.

PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. The interface for the blockchain is built using PHP, which generates the interface to take user input to display the result. The output is generated as HTML which is rendered by the Web Browser.

5.4.1 Cloud Analyst

CloudAnalyst, it is a tool which supports visual modeling and simulation of largescale applications that are deployed on Cloud Infrastructures. CloudAnalyst, built on top of CloudSim, allows description of application workloads, including information of geographic location of users generating traffic and location of data centers, number of users and data centers, and number of resources in each data center. Using this information, CloudAnalyst generates information about response time of requests, processing time of requests, and other metrics. By using CloudAnalyst, application developers or designers are able to determine the best strategy for allocation of resources among available data centers, strategies for selecting data centers to serve specific requests, and costs related to such operations.

5.5 Summary

Implementation is a process that describes the real world development of the system. In this chapter, the programming language used for the implementation of the system is discussed. The salient features of the language is talked about. It also describes the platform used to develop it. Also, the GUI for the system is described here as well as the various components of CloudSim is discussed here.

CHAPTER 6

TESTING

Program testing is a predominant verification and validation technique. Testing involves exercising the program using data like the real data processed by the program. The existence of program defects is inferred from unexpected system outputs. Testing may be carried out during the implementation phase to verify whether the software behaves as intended by its designer and after the implementation is complete. Testing presents an interesting anomaly for the software engineer. During earlier steps in design and implementation, the engineer attempts to build software from an abstract concept to a tangible implementation. The engineer creates a series of test cases that are intended to demolish the software that has been built. In fact, testing is the one step in software engineering process that could be viewed as destructive rather than constructive. Testing requires that the developer discard the preconceived notions of the correctness of the software just developed and overcome a “conflict of interest” that occurs when errors are uncovered.

6.1 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/ application/ product:

- Meets the requirements that guided its design and development
- Works as expected
- Can be implemented with the same characteristics
- And satisfies the needs of the stakeholders

Software testing, depending on the testing method employed, can be implemented at any time in the software development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

6.2 Testing Process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standard, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested. In some cases, one derives the test cases from the requirements of the product/software, which is to be developed.

6.2.1 Levels of Testing

Different levels of testing are used in testing process; each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing, and system and acceptance testing.

- **Unit Testing:** The first level of testing is called unit testing. Unit testing focuses on verification of the smallest unit of software design- the module. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essential for the verification of the code produced during the coding

phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. The unit test can be conducted in parallel for multiple modules.

- **Integration Testing:** The second level of testing is called integration testing. In this, many unit tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.
- **System Testing:** The third level of the testing is called system testing. In system testing, many subsystems are combined into entire working systems. The goal here is to see if the entire system works properly.

6.3 Unit Testing

Unit testing provides a sort of “living document”. Clients and other developers looking to learn how to use the module can look at the unit tests to determine how to use the module to fit their needs and gain a basic understanding of the API.

6.3.1 Unit Testing of Blockchain Records

Test Case ID	Unit Test Case 1
Description	To test the DNS blockchain, if it revokes the domain
Input	Domain records for revoking
Expected Output	Display of message regarding invalid record.
Actual Output	Obtained the error message
Remarks	Passed

Table 6.1 Unit Test Case 1

6.3.2 Unit Testing of CloudSim Module

Test Case ID	Unit test case 2
Description	To test the average response times over 24hrs
Input	Large number of cloudlets are assigned to the VM's (250 Cloudlets, 150 VMs) , (500 Cloudlets, 150 VMs), (750 Cloudlets, 150 VMs) , (1000 Cloudlets, 150 VMs)
Expected Output	Display of simulation graph
Actual Output	Simulation graph is obtained
Remarks	Passed

Table 6.2 Unit Test Case 2

Test Case ID	Unit test case 3
Description	To test the mean response times with respect to each User Base.
Input	Large number of cloudlets are assigned to the VM's (250 Cloudlets, 150 VMs) , (500 Cloudlets, 150 VMs), (750 Cloudlets, 150 VMs) , (1000 Cloudlets, 150 VMs)
Expected Output	Display of simulation graph
Actual Output	Simulation graph is obtained
Remarks	Passed

Table 6.3 Unit Test Case 3

Test Case ID	Unit test case 4
Description	To test the mean service time with respect to each data center.
Input	Large number of cloudlets are assigned to the VM's (250 Cloudlets, 150 VMs) , (500 Cloudlets, 150 VMs), (750 Cloudlets, 150 VMs) , (1000 Cloudlets, 150 VMs)
Expected Output	Display of simulation graph
Actual Output	Simulation graph is obtained
Remarks	Passed

Table 6.4 Unit Test Case 4

6.4 Integration Testing of Modules

The testing of combined parts of an application to determine if they function correctly together is integration testing.

Test Case ID	Integration Test Case 1
Description	1.The type of output is textual format 2. The simulation is run for specified number of times 3. Working of Algorithm with respect to the simulation
Input	Simulation file containing each userbase and data center
Expected Output	Assigning of DC for each User Base
Actual Output	Obtained the expected output
Remarks	Passed

Table 6.5: Integration Test Case 1

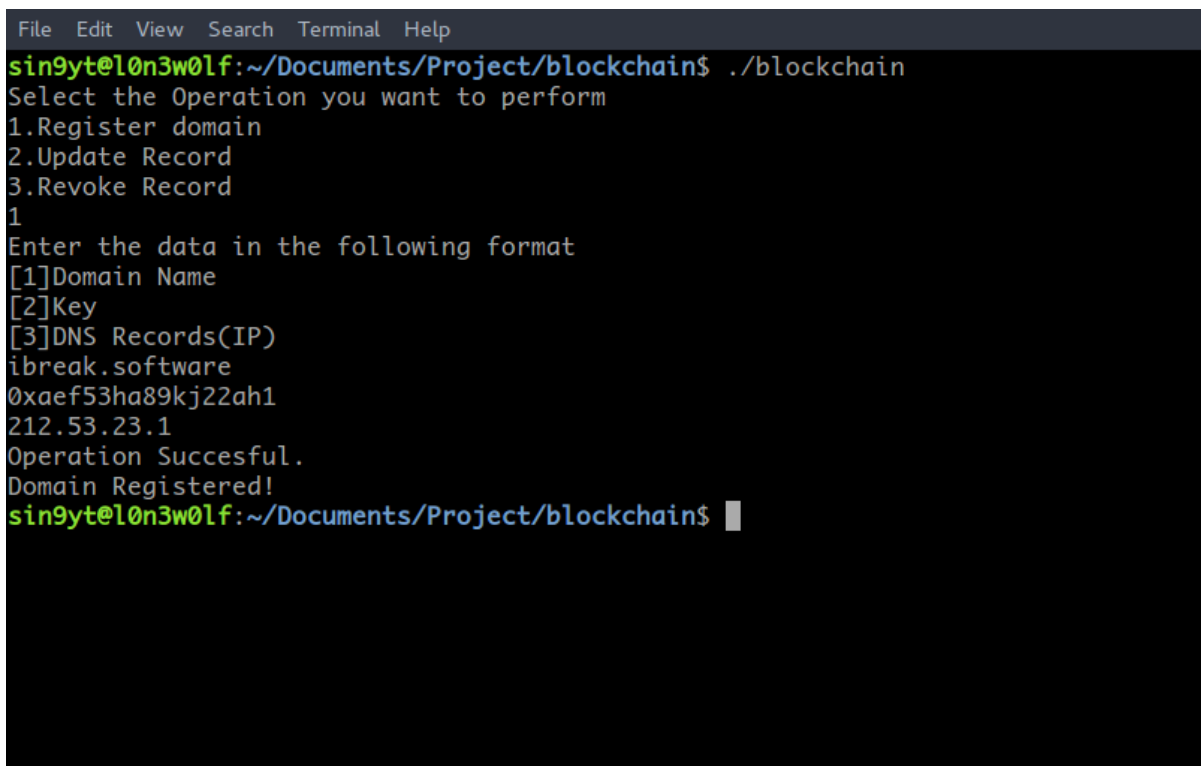
6.5 Summary

In this chapter, the discussion was mainly based on different software testing techniques. It covered different techniques like Unit Testing , Integration Testing and System Testing. This chapter also discussed the required results to be achieved at each level of testing. Various test cases with both expected and obtained output are compared.

CHAPTER 7

EXPERIMENTAL RESULTS

7.1 Block Chain



```
File Edit View Search Terminal Help
sin9yt@l0n3w0lf:~/Documents/Project/blockchain$ ./blockchain
Select the Operation you want to perform
1.Register domain
2.Update Record
3.Revoke Record
1
Enter the data in the following format
[1]Domain Name
[2]Key
[3]DNS Records(IP)
ibreak.software
0xae53ha89kj22ah1
212.53.23.1
Operation Successful.
Domain Registered!
sin9yt@l0n3w0lf:~/Documents/Project/blockchain$
```

Fig 7.1 Block Chain Interface

The blockchain interface lets you add, update or revoke records.

```

File Edit View Search Terminal Help
blockhash... b4e6d62310e3105c67c3b7b675a9ee11d99932bea58269344cce63dfdfd72ffe
datalen..... 32
data.....
                abc.com
                -xlkdfj
                19.9.9.9

-----

height..... 22
magic..... d5e8a97f
version..... 1
timestamp... 1493299798 (13:29:58 04/27/2017)
prevhash... b4e6d62310e3105c67c3b7b675a9ee11d99932bea58269344cce63dfdfd72ffe
blockhash... 8b44d5df9fbfad0f96698ec4abf2f72339d73a685f3c11848b05af9c8aa2fcab
datalen..... 54
data.....
                ibreak.software
                0xae53ha89kj22ah1
                212.53.23.1

-----

sin9yt@l0n3w0lf:~/Documents/Project/blockchain$

```

Fig 7.2 The output of all DNS records stored.

```

File Edit View Search Terminal Help
sin9yt@l0n3w0lf:~/Documents/Project/blockchain$ ./dumpindex
0   OFS: 0      LEN: 79
1   OFS: 79     LEN: 89
2   OFS: 168    LEN: 63
3   OFS: 231    LEN: 79
4   OFS: 310    LEN: 76
5   OFS: 386    LEN: 88
6   OFS: 474    LEN: 95
7   OFS: 569    LEN: 78
8   OFS: 647    LEN: 63
9   OFS: 710    LEN: 81
10  OFS: 791    LEN: 89
11  OFS: 880    LEN: 84
12  OFS: 964    LEN: 63
13  OFS: 1027   LEN: 78
14  OFS: 1105   LEN: 60
15  OFS: 1165   LEN: 79
16  OFS: 1244   LEN: 81
17  OFS: 1325   LEN: 64
18  OFS: 1389   LEN: 60
19  OFS: 1449   LEN: 63
20  OFS: 1512   LEN: 77
21  OFS: 1589   LEN: 99
sin9yt@l0n3w0lf:~/Documents/Project/blockchain$

```

Fig 7.3 Offset of Records

7.2 Cloud Simulation Results

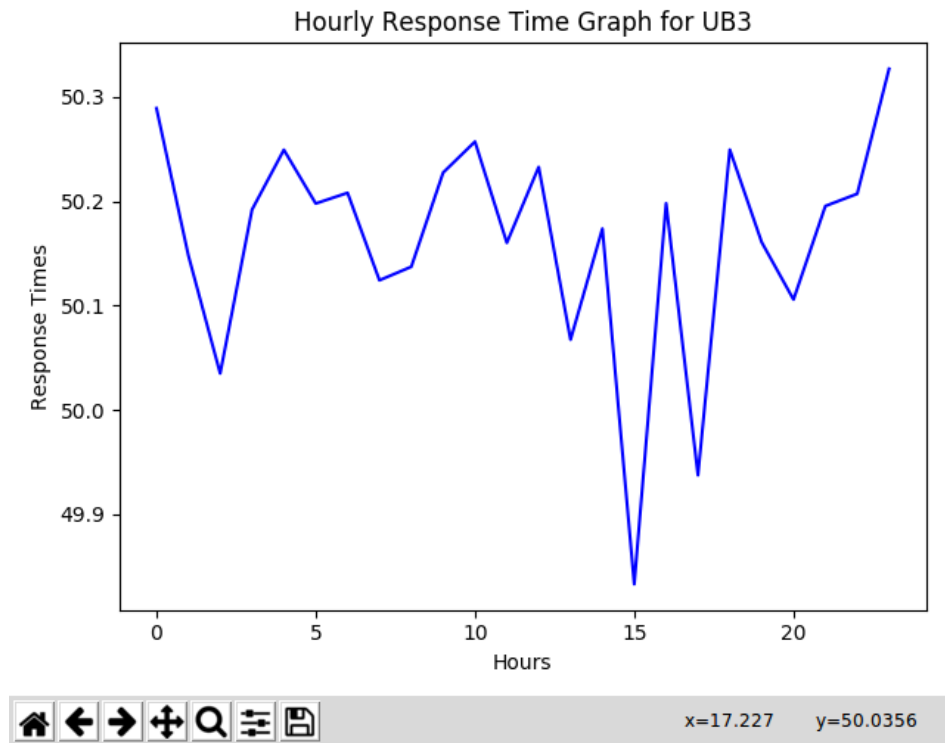


Fig 7.4 Hourly Response Time Graph

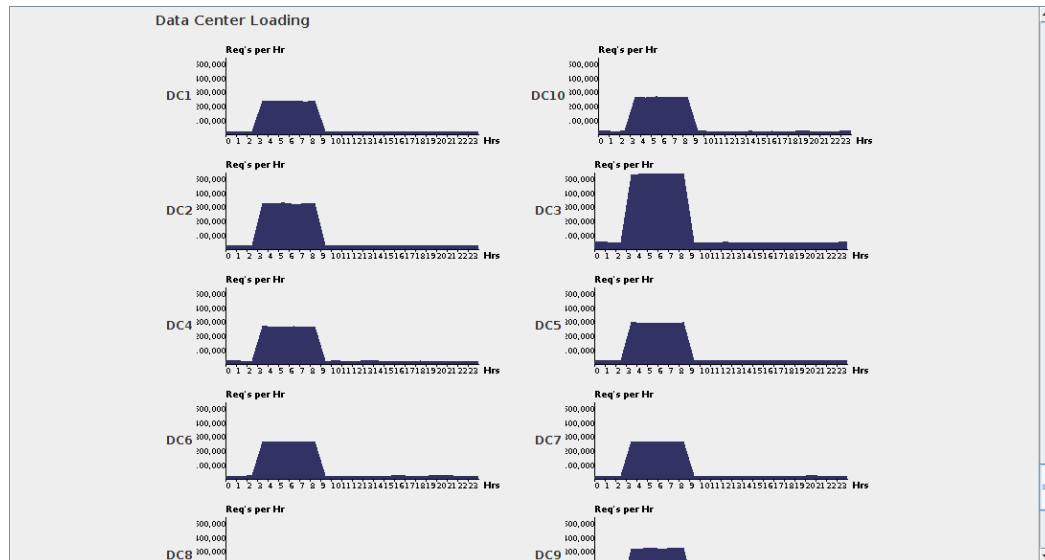
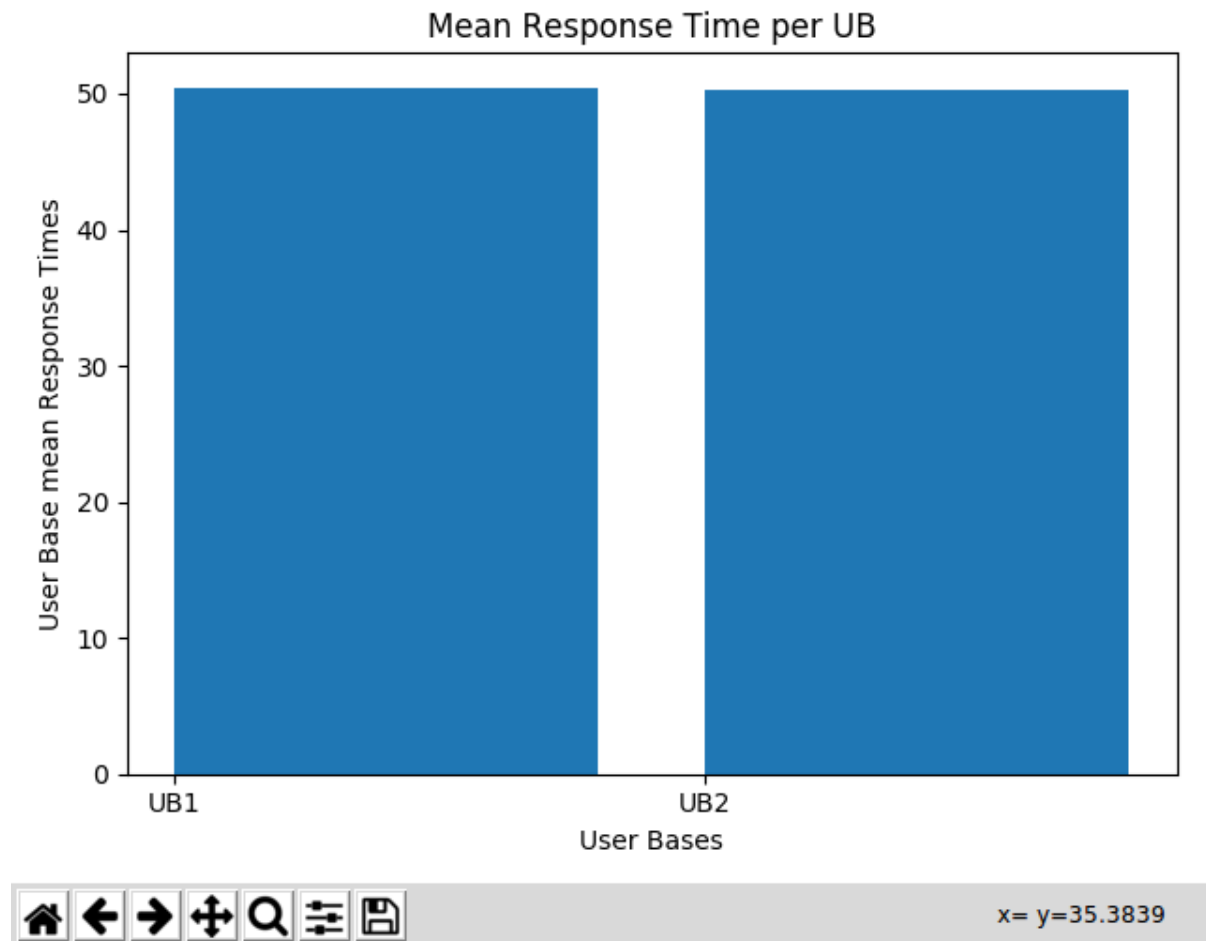
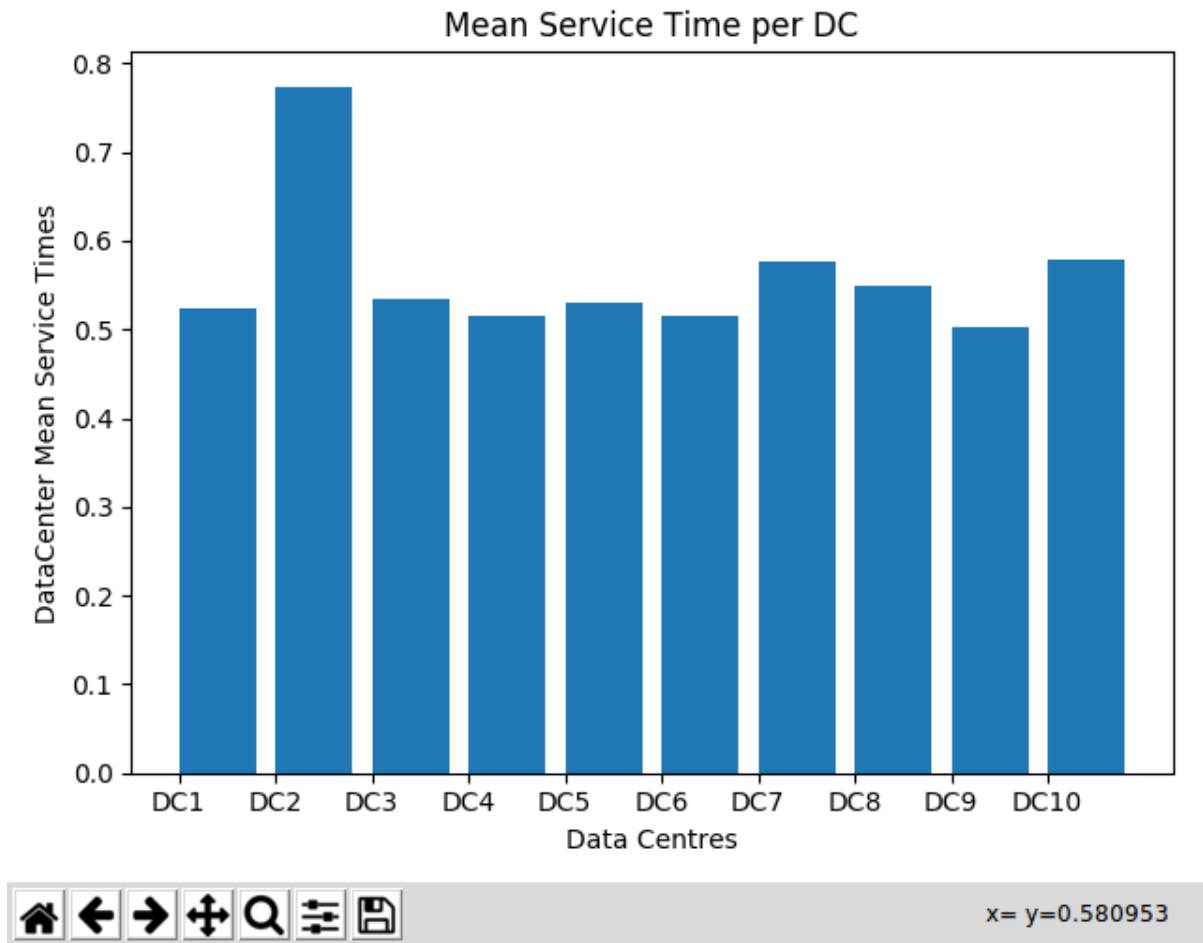


Fig 7.5 Loading Times

**Fig 7.6 User Base Mean Response Times**

**Fig 7.7 DataCenter Service Times**

7.3 Summary

This chapter shows all the results obtained by the system. Each section shows the output of each module. The blockchain section describes the interface and output obtained. The cloud simulation section shows various graphs obtained through unit testing.

CHAPTER 8

CONCLUSION

The motive of the project was to bring about a new architecture to the existing system of DNS. The new system was taken up to solve the current existing problem with the DNS. Along with it, a new algorithm is developed to help improve the times of DNS Resolution. The simulation was performed using a tool known as CloudSim. The new model of DNS was built using blockchain technology, which is currently popular with its use with BitCoin cryptocurrency. Blockchain helps in implementing the model in a decentralized and secure way. The algorithm for assigning best Trusted Entity was developed, which analyses different parameters to decide on the datacenter which takes least time for the resolution. The simulation models a real time scenario with large scale multiple users and datacenters. The algorithm is then executed over this simulation. Various graphs have been plotted to determine the parameters in developing the algorithm.

8.1 LIMITATIONS OF THE PROJECT

- The project is a simulation based model. Although, the simulation software is one of the standard platform used in the research but it again is a simulation.
- The model forces transparency for the DNS/PKI provider, but does not stop a malicious actor coercing data in TLD.
- The model parameters aren't as suitable as the requirement.

- The model is the first attempt to use an alternate architecture, so there is no or little help available.

8.2 FUTURE ENHANCEMENT

- The idea projected is going to be tested under different traffic load on different cloud server. The project has been studied under different condition on a simulation based environment called CloudSim. The real intricacies of the parameters measured would show up only when the model is subjected to real traffic on a cloud server.
- The model would be studied for if it can sustain under huge loads. It is an effort to create a scalable system.
- A system to cross check the data in Trusted Entities by enabling tracking, but it comes under the cost of reduced performance.

REFERENCES

- [1] Francesca Musiani, A Decentralized Domain Name System? User-Controlled Infrastructure as Alternative Internet Governance, MA:The MIT Press.
- [2] Harry Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, Arvind Narayanan, An empirical study of Namecoin and lessons for decentralized namespace design, Princeton University.
- [3] Aaron Wright, Primavera De Filippi, Decentralized blockchain technology and the rise of lex cryptographia, intGovt Forum.
- [4] Christopher Allen, Arthur Brock, Vitalik Buterin, Decentralized Public Key Infrastructure, Web Of Trust.
- [5] Venugopalan Ramasubramanian, Emin Gün Sirer, The Design and Implementation of a Next Generation Name Service for the Internet, Cornell University.
- [6] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Rajkumar Buyya, CloudSim a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms , John Wiley & Sons, 2014
- [7] E. Cohen and H. Kaplan. Proactive Caching of DNS Records: Addressing a Performance Bottleneck. Symposium on Applications and the Internet, San Diego Mission Valley CA, Jan 2001.
- [8] P. Thurrott. Microsoft Suffers Another DoS Attack.
www.winnetmag.com/WindowsSecurity/Article/ArticleID/19770/Window ,Jan 2001.

TEAM INFORMATION

<p>1. NAME: Arisha Siddiqui</p> <p>USN: 1PE13CS032</p> <p>CONTACT NUMBER: +91 97422 41753</p> <p>EMAIL ID: arisha.siddiqui1994@yahoo.com</p> <p>ADDRESS: #1205, Suraksha Homes, 22nd Cross, Sector 3, HSR Layout,Bangalore-560102, Karnataka</p>	
<p>2. NAME: Shashish Jha</p> <p>USN: 1PE13CS139</p> <p>CONTACT NUMBER: +91 74068 80137</p> <p>EMAIL ID: shashishjha81@gmail.com</p> <p>ADDRESS: PESIT-BSC Boy's Hostel, Konnapana Agrahara , Hosur Road , 1KM from Electronic City , Bangalore -560100</p>	
<p>3. NAME: Sindhoor Tilak S Hegde</p> <p>USN: 1PE13CS148</p> <p>CONTACT NUMBER: +91 9686642561</p> <p>EMAIL ID: sindhoorhegde378@gmail.com</p> <p>ADDRESS: PESIT-BSC Boy's Hostel, Konnapana Agrahara , Hosur Road , 1KM from Electronic City , Bangalore -560100</p>	
<p>4. NAME: Venugopala B.V</p> <p>USN: 1PE14CS431</p> <p>CONTACT NUMBER: +91 84539 18115</p> <p>EMAIL ID: venu.bvmsd@gmail.com</p> <p>ADDRESS: Sai Balaji PG, Krishna Reddy Layout, Electronic City Phase 2, Bangalore-560100</p>	