B. Tech Project Work Report

On

Public Facing DNS Resolver for NITK

Chirag R

(201CS170)

Akash Prasad

(201CS205)

Attada Ramprasad

(201CS210)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE - 575025

August, 2025

DECLARATION

We hereby declare that the B. Tech Project Work Report entitled **PUBLIC FACING DNS RESOLVER FOR NITK** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology** in **Computer Science and Engineering** is a *bonafide report of the work carried out by me*. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

Register Number, Name & Signature of the Student:

1. 201CS170, Chirag R
2. 201CS205, Akash Prasad
3. 201CS210, Attada Ramprasad

Department of Computer Science and Engineering

Place: NITK, Surathkal.

Date:

CERTIFICATE

This is to *certify* that the B. Tech Project Work Report entitled **PUBLIC FACING DNS RESOLVER FOR NITK** submitted by:

Sl.No. Register Number & Name of Student(s)

(1) 201CS170 Chirag R

(2) 201CS205 Akash Prasad

(3) 201CS210 Attada Ramprasad

as the record of the work carried out by them, is *accepted as the B. Tech Project Work Report submission* in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in Computer Science and Engineering.

Guide

Dr. Saumya Hegde

(Name and Signature with Date)

Chairman - DUGC

(Signature with Date and Sea;)

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our guide Dr. Saumya Hegde, for her invaluable guidance and expertise throughout the project. Her valuable insights, continuous encouragement, and patient mentoring have been instrumental in shaping this thesis and enriching our understanding of the subject matter. We are truly grateful for her dedication and the time she has invested in supervising this project.

We are also indebted to the Head of the Department, Dr. Manu Basavaraju, for his support and encouragement throughout our academic journey. His visionary leadership and commitment to excellence have created a conducive environment for learning and research in our department.

We would like to extend our thanks to the faculty members of the Department of Computer Science and Engineering for their insightful lectures, stimulating discussions, and valuable feedback, which have broadened our knowledge and shaped our approach.

We would like to express our gratitude to our fellow classmates and friends who have provided constant motivation, encouragement, and support during this challenging yet rewarding journey. Their camaraderie and shared experiences have made this academic pursuit all the more meaningful.

Abstract

This project introduces the steps involved in the development of a public-facing DNS server dedicated to query resolution. The project commenced with the construction of a local DNS authoritative server using BIND, which was transformed into a recursive query resolver. Our server is supposed to provide efficient resolution of domain names for external clients. Leveraging robust architecture it offers rapid response times and dependable performance. Furthermore, our implementation will also prioritize adherence to industry standards, ensuring compatibility and seamless integration with existing DNS infrastructures. This DNS server serves as a valuable resource, enhancing the online experience by providing reliable domain name resolution services

Keywords: DNS server, Authoritative name server, Query resolution, Recursive resolver, BIND, performance optimization, compatibility

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Domain Name System (DNS) is a hostname for IP address translation service. DNS is a distributed database implemented in a hierarchy of name servers. It is an application layer protocol for message exchange between clients and servers. It is required for the functioning of the Internet.

Every host is identified by the IP address but remembering numbers is very difficult for people also the IP addresses are not static therefore a mapping is required to change the domain name to the IP address. So DNS is used to convert the domain name of the websites to their numerical IP address. It is very difficult to find out the IP address associated with a website because there are millions of websites and with all those websites we should be able to generate the IP address immediately, there should not be a lot of delays for that to happen organization of the database is very important.

DNS records, Domain name, IP address their validity, life time, and all the information related to that domain name. These records are stored in a tree-like structure. Namespace is the Set of possible names, flat or hierarchical. The naming system maintains a collection of bindings of names to values – given a name, a resolution mechanism returns the corresponding value. Name server is an implementation of the resolution mechanism.

1.0.1 Name to Address Resolution

The host requests the DNS name server to resolve the domain name. And the name server returns the IP address corresponding to that domain name to the host so that the host can future connect to that IP address. 1.1 shows the name to address

resolution process.

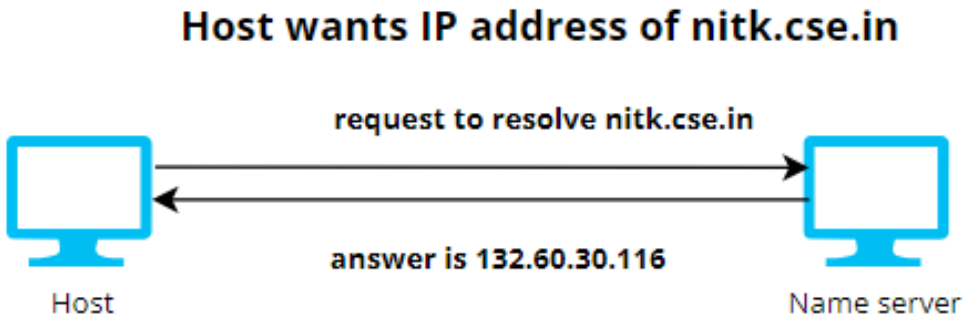

Figure 1.1: Name to Address Resolution

Hierarchy of Name Servers Root name servers: It is contacted by name servers that can not resolve the name. It contacts the authoritative name server if name mapping is not known. It then gets the mapping and returns the IP address to the host.

Top-level domain (TLD) server: It is responsible for com, org, edu, etc, and all top-level country domains like uk, fr, ca, in, etc. They have info about authoritative domain servers and know the names and IP addresses of each authoritative name server for the second-level domains.

Authoritative name servers are the organization's DNS servers, providing authoritative hostnames to IP mapping for organization servers. It can be maintained by an organization or service provider. In order to reach nitk.cse.in we have to ask the root DNS server, then it will point out to the top-level domain server and then to the authoritative domain name server which actually contains the IP address. So the authoritative domain server will return the associative IP address.

1.0.2 Domain Name Server

The client machine sends a request to the local name server, which, if the root does not find the address in its database, sends a request to the root name server, which in turn, will route the query to a top-level domain (TLD) or authoritative name server.

The root name server can also contain some hostName to IP address mappings. The Top-level domain (TLD) server always knows who the authoritative name server is. So finally the IP address is returned to the local name server which in turn returns the IP address to the host. 1.2 summarizes this process.

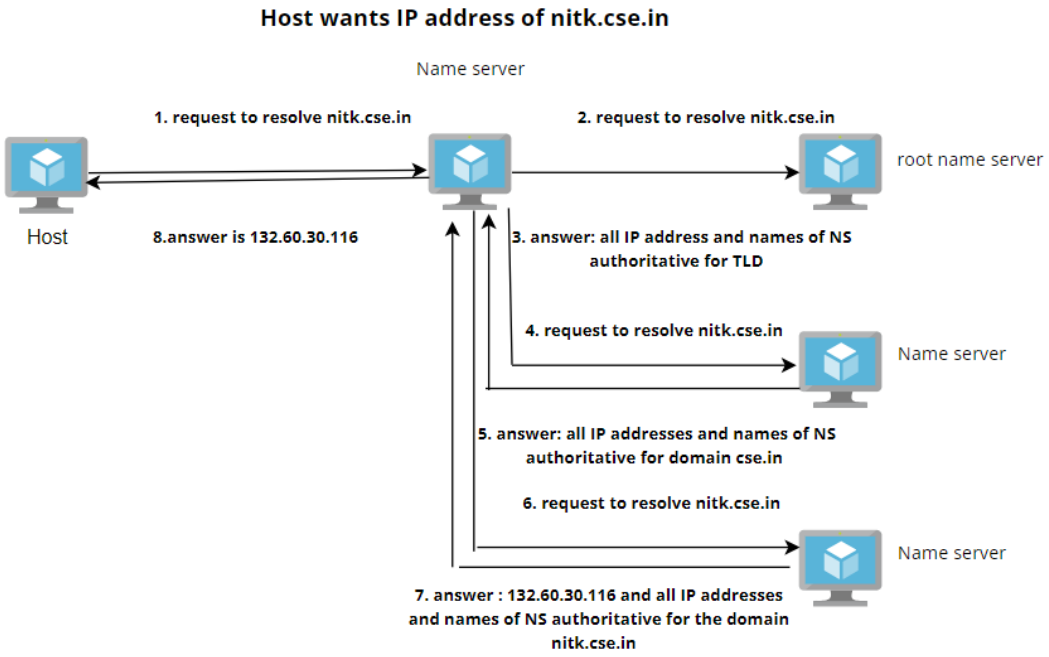

Figure 1.2: Domain Name Server

1.0.3 Working of a DNS Server

The working of DNS starts with converting a hostname into an IP Address. A domain name serves as a distinctive identification for a website. It is used in place of an IP address to make it simpler for consumers to visit websites. Domain Name System works by executing the database whose work is to store the name of hosts which are available on the Internet. The top-level domain server stores address information for top-level domains such as .com and .net, .org, and so on. If the Client sends the request, then the DNS resolver sends a request to DNS Server to fetch the IP Address. In case, when it does not contain that particular IP Address with a hostname, it forwards the request to another DNS Server. When IP Address has arrived at the resolver, it completes the request over Internet Protocol.

Chapter 2

Literature Survey

2.0.1 Performance Benefits

As web pages become more complex, referencing resources from numerous domains, DNS lookups can become a significant bottleneck in the browsing experience. Whenever a client needs to query a DNS resolver over the network, the latency introduced can be significant, depending on the proximity and number of name servers the resolver has to query (more than 2 is rare, but it can happen). As an example, the following screen shot in 2.1 as per Google (2018) shows the timings reported by the Page Speed web performance measurement tool. Each bar represents a resource referenced from the page; the black segments indicate DNS lookups. In this page, 13 lookups are made in the first 11 seconds in which the page is loaded. Although several of the lookups are done in parallel, the screen shot shows that 5 serial lookup times are required, accounting for several seconds of the total 11 seconds page load time.

There are two components to DNS latency: **Latency between the client (user)** and **DNS resolving server**. In most cases this is largely due to the usual round-trip time (RTT) constraints in networked systems: geographical distance between client and server machines; network congestion; packet loss and long retransmit delays (one second on average); overloaded servers, denial-of-service attacks and so on. Latency between resolving servers and other name servers. This source of latency is caused primarily by the following factors:

Cache misses. If a response cannot be served from a resolver's cache, but requires recursively querying other name servers, the added network latency is considerable, especially if the authoritative servers are geographically remote.

Underprovisioning. If DNS resolvers are overloaded, they must queue DNS res-

olution requests and responses, and may begin dropping and retransmitting packets.

Malicious traffic. Even if a DNS service is overprovisioned, DoS traffic can place undue load on the servers. Similarly, Kaminsky-style attacks can involve flooding resolvers with queries that are guaranteed to bypass the cache and require outgoing requests for resolution.

According to Google, cache miss factor is the most dominant cause of DNS latency and let us see more about it.

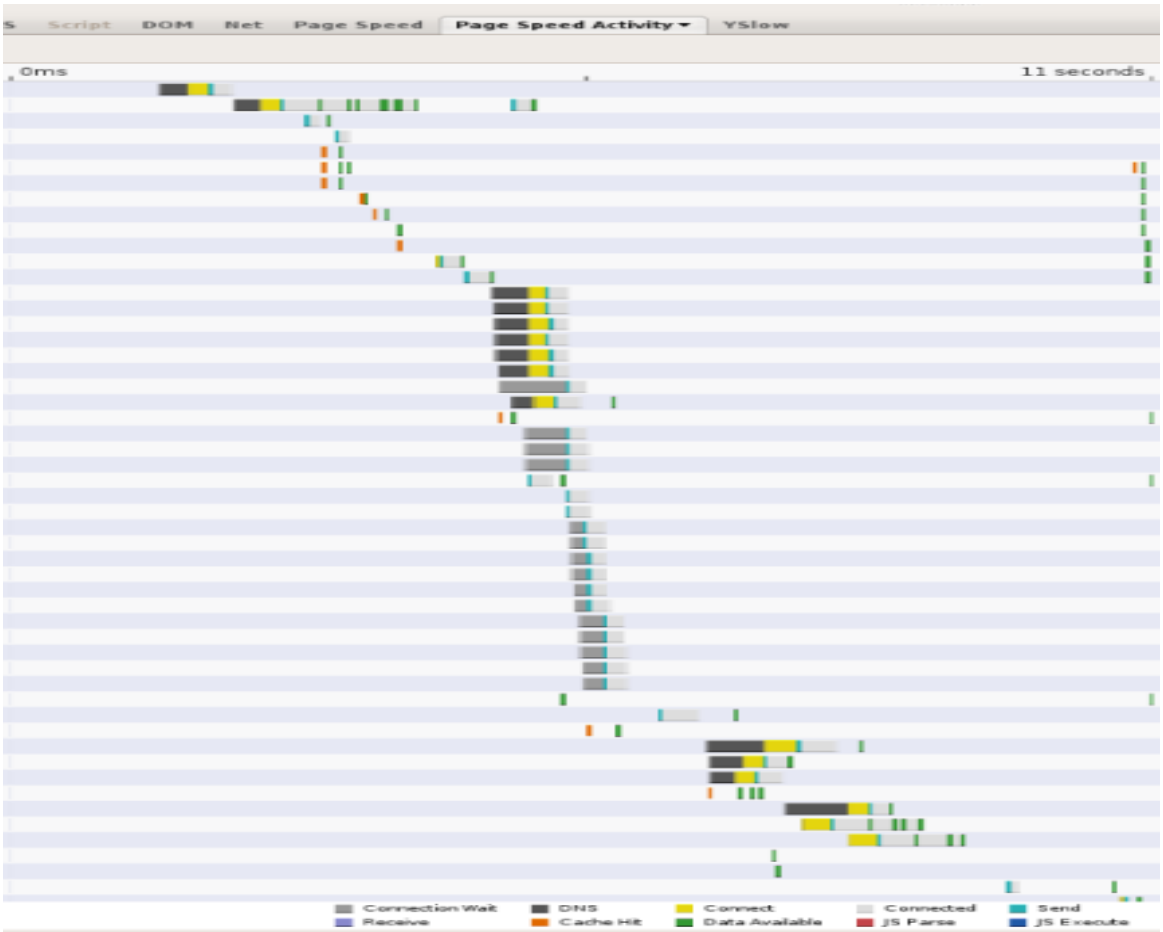

Figure 2.1: DNS Latency of Google's Public facing DNS Resolver

A Cache Misses

Even if a resolver has abundant local resources, the fundamental delays associated with talking to remote name servers are hard to avoid. In other words, assuming the resolver is provisioned well enough so that cache hits take zero time on the server-side, cache misses remain very expensive in terms of latency. To handle a miss, a

resolver has to talk to at least one, but often two or more external name servers. Operating the Googlebot web crawler, we have observed an average resolution time of 130 ms for name servers that respond. However, a full 4-6 percent of requests simply time out, due to UDP packet loss and servers being unreachable. If we take into account failures such as packet loss, dead name servers, DNS configuration errors, etc., the actual average end-to-end resolution time is 300-400 ms. However, there is high variance and a long tail.

Though the cache miss rate may vary among DNS servers, cache misses are fundamentally difficult to avoid, for the following reasons:

Internet size and growth. Quite simply, as the Internet grows, both through the addition of new users and of new sites, most content is of marginal interest. While a few sites (and consequently DNS names) are very popular, most are of interest to only a few users and are accessed rarely; so the majority of requests result in cache misses.

Low time-to-live (TTL) values. The trend towards lower DNS TTL values means that resolutions need more frequent lookups.

Cache isolation. DNS servers are typically deployed behind load balancers which assign queries to different machines at random. This results in each individual server maintaining a separate cache rather than being able to reuse cached resolutions from a shared pool.

2.0.2 Risks Associated

A The alleged Public nature of DNS Data

It has long been claimed that "the data in the DNS is public".While this sentence makes sense for an Internet-wide lookup system, there are multiple facets to the data and metadata involved that deserve a more detailed look. First, access control lists and private namespaces notwithstanding, the DNS operates under the assumption that public-facing authoritative name servers will respond to "usual" DNS queries for any zone they are authoritative for without further authentication or authorization of the client (resolver). Due to the lack of search capabilities, only a given QNAME will reveal the resource records associated with that name (or that name's non- existence). In other words: one needs to know what to ask for, in order to receive a response.

The zone transfer QTYPE is often blocked or restricted to authenticated/authorized access to enforce this difference (and maybe for other reasons).

Another differentiation to be considered is between the DNS data itself and a particular transaction (i.e., a DNS name lookup). DNS data and the results of a DNS query are public, within the boundaries described above, and may not have any confidentiality requirements. However, the same is not true of a single transaction or a sequence of transactions; that transaction is not / should not be public. A typical example from outside the DNS world is: the web site of Alcoholics Anonymous is public; the fact that you visit it should not be.

B Data in the DNS Request

The DNS request includes many fields, but two of them seem particularly relevant for the privacy issues: the QNAME and the source IP address. "source IP address" is used in a loose sense of "source IP address + maybe source port", because the port is also in the request and can be used to differentiate between several users sharing an IP address.

The QNAME is the full name sent by the user. It gives information about what the user does ("What are the MX records of example.net?" means he probably wants to send email to someone at example.net, which may be a domain used by only a few persons and is therefore very revealing about communication relationships). Some QNAMEs are more sensitive than others. For instance, querying the A record of a well-known web statistics domain reveals very little (everybody visits web sites that use this analytics service), but querying the A record of www.verybad.example where verybad.example is the domain of an organization that some people find offensive or objectionable may create more problems for the user. Also, sometimes, the QNAME embeds the software one uses, which could be a privacy issue.

Another important thing about the privacy of the QNAME is the future usages. Today, the lack of privacy is an obstacle to putting potentially sensitive or personally identifiable data in the DNS. At the moment, your DNS traffic might reveal that you are doing email but not with whom. If your Mail User Agent (MUA) starts looking up Pretty Good Privacy (PGP) keys in the DNS [DANE-OPENPGPKEY], then privacy becomes a lot more important. And email is just an example; there would be other

really interesting uses for a more privacy- friendly DNS.

For the communication between the stub resolver and the recursive resolver, the source IP address is the address of the user's machine. Therefore, all the issues and warnings about collection of IP addresses apply here. For the communication between the recursive resolver and the authoritative name servers, the source IP address has a different meaning; it does not have the same status as the source address in an HTTP connection. It is now the IP address of the recursive resolver that, in a way, "hides" the real user. However, hiding does not always work. Sometimes [CLIENT-SUBNET] is used. Sometimes the end user has a personal recursive resolver on her machine. In both cases, the IP address is as sensitive as it is for HTTP

A note about IP addresses: there is currently no IETF document that describes in detail all the privacy issues around IP addressing. In the meantime, the discussion here is intended to include both IPv4 and IPv6 source addresses. For a number of reasons, their assignment and utilization characteristics are different, which may have implications for details of information leakage associated with the collection of source addresses. (For example, a specific IPv6 source address seen on the public Internet is less likely than an IPv4 address to originate behind a CGN or other NAT.) However, for both IPv4 and IPv6 addresses, it's important to note that source addresses are propagated with queries and comprise metadata about the host, user, or application that originated them.

C In the Servers

The DNS servers (recursive resolvers and authoritative servers) are enablers: they facilitate communication between an initiator and a recipient without being directly in the communications path. As a result, they are often forgotten in risk analysis. But, to quote again [RFC6973], "Although enablers may not generally be considered as attackers, they may all pose privacy threats (depending on the context) because they are able to observe, collect, process, and transfer privacy-relevant data." Sometimes, enablers become observers when they start collecting data.

Many programs exist to collect and analyze DNS data at the servers – from the "query log" of some programs like BIND to tcpdump and more sophisticated programs like PacketQ and DNSmezzo. The organization managing the DNS server can use this

data itself, or it can be part of a surveillance program like PRISM [prism] and pass data to an outside observer.

Sometimes, this data is kept for a long time and/or distributed to third parties for research purposes [ditl] [day-at-root], security analysis, or surveillance tasks. These uses are sometimes under some sort of contract, with various limitations, for instance, on redistribution, given the sensitive nature of the data. Also, there are observation points in the network that gather DNS data and then make it accessible to third parties for research or security purposes ("passive DNS" [passive-dns]).

Apart from these, K. Jerabek et. al. (2023) may help to understand the configurations that can be used when using DoH packets. J. Park et. al. (2019) discusses security and inaccuracies posed by DNS resolvers. This can help to identify places where we might make mistakes and rectify them. Yingdi Tu et. al. (2012) examines how DNS caching resolvers select authoritative name server to send a query to since operators of high-profile DNS zones use multiple authority servers for performance and robustness. Marc Kührer et. al. (2015) mainly talks about the response authenticity and integrity of all open DNS resolvers in the entire IPv4 address space and says that the DNS protocol does not only have flaws at the network-level in terms of traffic amplification vulnerabilities but also lack verification mechanisms at the application-level to sufficiently protect end hosts from malicious resolvers that redirect clients to suspicious content.

Chapter 3

Problem Statement

The main objective of our project, as the name suggests, is to build a public facing DNS resolver for our college, National Institute of Technology Karnataka, Surathkal(NITK). As shown in a research paper by Antonia Affinito et. al. (2022), local DNS servers show better performance than public ones(Google, OpenDNS). This is in addition to better security measures in place, due to the factor that new and improved security implementations can be tested and implemented early.

In addition to this, we can also implement new methods in place, such as caching strategies to store DNS records in intermediate servers so that the delay in DNS lookup can be minimised, load balancing, as well as efficient query handling mechanisms.

Along with this, we can also customise the local DNS server setup according to the needs of the institute, at least in the initial stages of its development. In this aspect, the resolver can be developed in three stages. The first stage will be the local prototype setup, which can be done on local systems, either thorugh an online server or on a physical system through a virtualbox system in place. The second stage will be to implement the resolver on the institute level so that the students and the staff of NITK can use it over the institute WiFi. The final stage will be to get the product out into the world as a public resolver for the benefit of all the people.

It would also be a great experience for the students as well as the faculty involved to understand the working of DNS, and the network and the security practices that are invloved with it. Since this project is mostly going to be a multi semester one, the next batches can continue working on the project, and hopefully, the public DNS server will be deployed sometime in the future.

There are some things we need to take care of in the process. The first one

is the security of the DNS server. A DNS resolver is vulnerable to many kinds of attacks, which includes DNS hijacking and DNS flood attacks. Thus, the best security practices in the market have to be researched and implemented to safeguard the developed system. This might be out of the scope in our phase of the project though. The future teams can try working on it.

We will also have to consider the legal aspects such as data privacy laws and copyright issues. This comes into place when we try to implement caching mechanism in place, as we should make sure that we are not misusing the data that is being stored. We also have to make sure that we are following the terms of service that is provided by the public DNS resolvers.

Finally, the working of the DNS servers have to be studied in depth so that relevant standards and best practices are being followed in our project. To implement it after the first step, we will need to be in constant communication with CCC to obtain dockers and for following the terms of NITK institute. Marc Kührer et. al. (2015) tells us that the majority of the resolvers that disclose their DNS server software operate on BIND. This gives us the inspiration to use BIND for our setup.

Chapter 4

Experimental Setup

The primary setup consists a set of virtual machines (VMs) set up in a system in M TECH Lab. Since physical access to the system was not available, all work was done using SSH remotely. Due to SSH access to the system being command based, Vagrant, a source-available software product for building and maintaining portable virtual software development environments was used to set up and configure the VMs.

4.0.1 Setting up the VMs

Initally Ubuntu servers were unreachable and verification certificate failed. To resolve this, changed deb and deb-src to the latest mirror files in source.list file of Ubuntu in the host machine. Also, NITK captive portal was not logged in. To log in through command line used wget post command. Ubuntu servers were available after this and created the VMs using vagrant (Ubuntu 22.0.4 by default). Vagrantfile is a script that allows us to create multiple VMs and add all the specifications and configurations needed such as the memory that needs to be allocated, the total number of CPUs needed, IP addresses and all softwares (in our case BIND name server software - BIND9, a suite of software for interacting with the DNS) to be installed on each VM. 5 VMs were configured with names machine1, machine2, machine3, machine4 and machine5 each with 4GB RAM and 3 CPUs. They were configured with IP addresses given in table 4.1. We have used the same subnet - 192.168.56.0/24. The configuration is set only for IPv4 now. After this the VMs are ready to be logged in through Vagrant.

4.0.2 Authoritative Nameserver

The first step was to create a private network to improve the management of servers at NITK. This was done by setting up an internal DNS server using two virtual machines. BIND9 was used to resolve private hostnames and IP addresses. machine1 was used to serve as the Primary DNS Server while the machine2 as the Secondary DNS server (DNS name servers). machine3 and machine4 were configured to act as client servers. As shown in the table 4.1, all servers are connected to a project that runs on example.com with naming schemes based around the subdomain nitk.example.com to refer to the private subnet or zone. FQDN refers to Fully Qualified Domain Name and is the complete domain name for a specific computer, or host, on the internet. The DNS servers will attempt to route requests internally first before trying to reach them on the public Internet.

Table 4.1: Details of VMs used as Servers

Host VM	Role	Private FQDN	Private IP Address
machine1	Primary Nameserver	machine1.nitk.example.com	192.168.56.2
machine2	Secondary Nameserver	machine2.nitk.example.com	192.168.56.3
machine3	Client Server 1	machine3.nitk.example.com	192.168.56.10
machine4	Client Server 2	machine4.nitk.example.com	192.168.56.11

A Primary nameserver options

Configuration of BIND is mainly done at named.conf which includes multiple other configuration files. These files begin with named as it is the process that BIND runs on (name daemon). Configuring the primary server name server is started by modifying this file. IN named.conf.options, we create a new Access Control List (ACL) block called "trusted". It consists of IP addresses of clients from whom recursive DNS queries are allowed - the VMs that we have created. Now, in the options block, we add configurations to enable recursive queries from "trusted" clients, listen on the IP address it was configured with, disable zone transfers by default and enabling forwarders to 8.8.8.8 and 8.8.4.4 which are Google's public DNS servers. Forwarders

are used to reduce load on external nameservers and allow queries by servers that do not have direct access to the Internet. Any public recursive DNS resolver could be used. These configurations ensure that only the trusted servers can query the primary DNS server for external domains.

B Zones and zone files

Now, we specify the DNS zones by configuring the named.conf.local file where we mention our forward and revere zones. They help in defining and managing the DNS records for a given scope. The forward zone will be nitk.example.com as the domains defined are in this subdomain. The zone file path that will be created is given along with an allow-transfer option to the secondary server's IP address. Similarly as the defined IP addresses are in the 192.168.56.0/24 IP space, the reverse zone will define reverse lookups in the range. Additional zone and zone files for each subnet may be given here as given above.

Foward zone path defines DNS records for forward DNS lookups. Directory for the zone files are created. BIND has a sample db.local zone file on which the forward zone file can be based. Its copy is made and edited to configure the forward DNS records, both NS and A records for the name server and client servers respectively in a similar format. Reverse zone files define DNS PTR records for reverse DNS lookups. Procedure followed is similar to that of forward zone files. db.127 is the sample file which is used to store information for the local loopback interface. The reverse zone file created is db.192. Instead of A records, PTR records are added for all servers are added here. Once errors were checked for, BIND was restarted to finish seting up the primary DNS servers.

C Secondary nameserver

A secondary DNS server is also configured that responds to requests if primary server is unavailable. named.conf.options file is configured as was done in case of primary server with the server listening on the IP address assigned to it. Secondary zones are also configured similarly except that the type is slave, file does not contain a path, and a masters directive is set to the primary DNS server's private IP address. After checking the validity of the configured files, BIND is restarted and secondary server

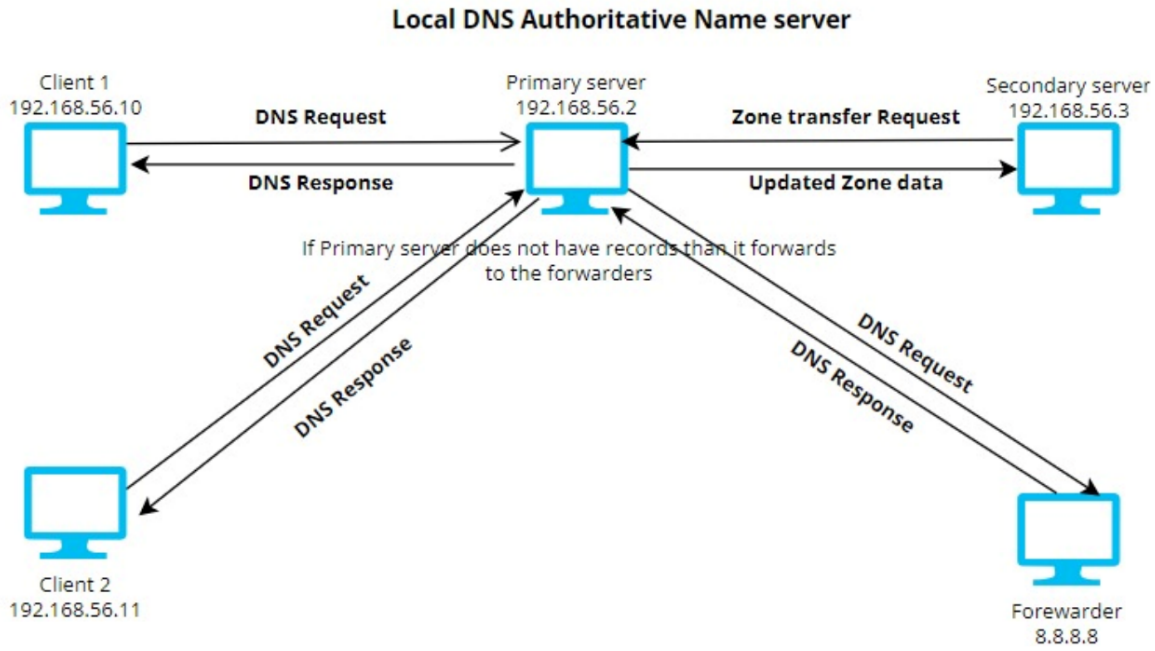

Figure 4.1: Query Resolution in the Setup

is up.

D Client servers

The client servers need to be configured to use machine1 and machine2 as name servers. The network interface that uses the same subnet is eth1. Netplan is a utility for easily configuring networking on a Linux system where a description of required interfaces and its definition can be created. A YAML file for Netplan configuration is written that tells the network interface and namespace addresses along with the DNS zone. This new configuration is applied and can be checked to see the system's DNS resolver. With this setup, query resolution happens as shown in 4.1.

4.0.3 Recursive Resolver

The next part of the project aims to add a recursive resolver to the authoritative nameserver configured. It was learnt that just setting recursion to yes and allow recursion as well as removing forwarders from options should be enough to create a recursive resolver. This is because forwarders are used to resolve queries when records are not found in the nameserver. If forwarders are not present, the resolver falls back to recursion, querying the root, TLD and authoritative nameserver.

In order to verify that recursion is being used and not forwarding, the client

requesting the resolution is removed from the "trusted" Access Control List (ACL). If recursion is being used, the nameserver will not be able to resolve the query names since only the IP addresses present in the "trusted" ACL are allowed to perform recursion as configured.

nslookup command is used to test whether the clients can query the name servers for both forward and reverse lookups.

Chapter 5

Results and Analysis

After making all configurations and resolving errors, an authoritative nameserver was built. Checking the DNS resolver related interfaces on machine4, a client server, we find the output in figure 5.1 showing the eth1 interface with the IP addresses of primary (machine1) and secondary (machine2) servers as the DNS servers.

Requesting machine3 on machine4 server using the nslookup command gives the output as shown in figure 5.2. This is because machine3 expands to become machine3.nitk.example.com and this name is looked on the subdomain before anywhere else.

The primary and secondary servers may be queried specifically by adding their IP addresses to the nslookup command to resolve the IP address of machine3. The output is shown in figure 5.3 and figure 5.4 respectively. This shows that authoritative nameservers are resolving queries that have records in the forward zone files.

Even reverse lookups can be queried providing the IP addresses of different machines defined in reverse zone files for the required servers as shown in figure 5.5. Hence reverse lookups utilizing reverse zone files are also working as expected.

Now, to test whether recursive queries are resolved, an external domain name google.com was queried. The corresponding IP address was resolved and returned as shown in figure 5.6. This may be due to the forwarders present in the resolver. Since the records cannot be found in the zone files, the query is redirected to the forwarders who resolve the IP address and returns it.

These forwarders were disabled by commenting out the servers. On querying again, a similar output was produced as given in figure 5.7. However, this time, as forwarders were not part of the system, the only way for resolution to happen is

```
vagrant@machine4:~$ sudo resolvectl
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub

Link 2 (eth0)
    Current Scopes: DNS
         Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 10.0.2.3
       DNS Servers: 10.0.2.3
        DNS Domain: nitk.ac.in

Link 3 (eth1)
    Current Scopes: DNS
         Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 192.168.56.2
       DNS Servers: 192.168.56.2 192.168.56.3
        DNS Domain: nitk.example.com
```

Figure 5.1: Output of resolvectl command

```
vagrant@machine4:~$ nslookup machine3
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   machine3.nitk.example.com
Address: 192.168.56.10
```

Figure 5.2: Output of nslookup command

```
vagrant@machine4:~$ nslookup machine3 192.168.56.2
Server:         192.168.56.2
Address:        192.168.56.2#53

Name:   machine3.nitk.example.com
Address: 192.168.56.10
```

Figure 5.3: Output of nslookup command for Primary Nameserver

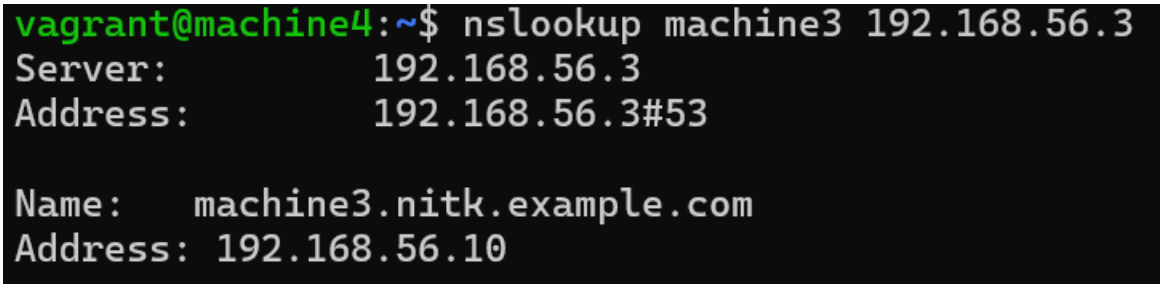

Figure 5.4: Output of nslookup command for Secondary Nameserver

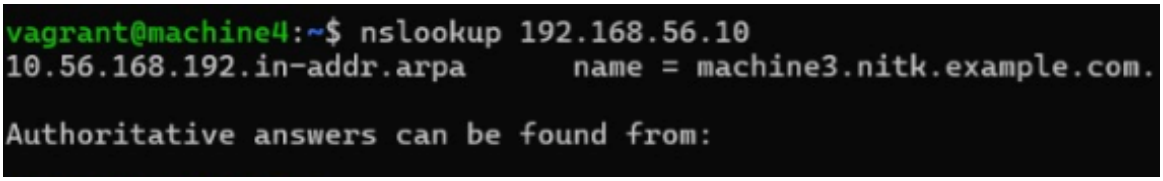

Figure 5.5: Output of nslookup command with IP Address

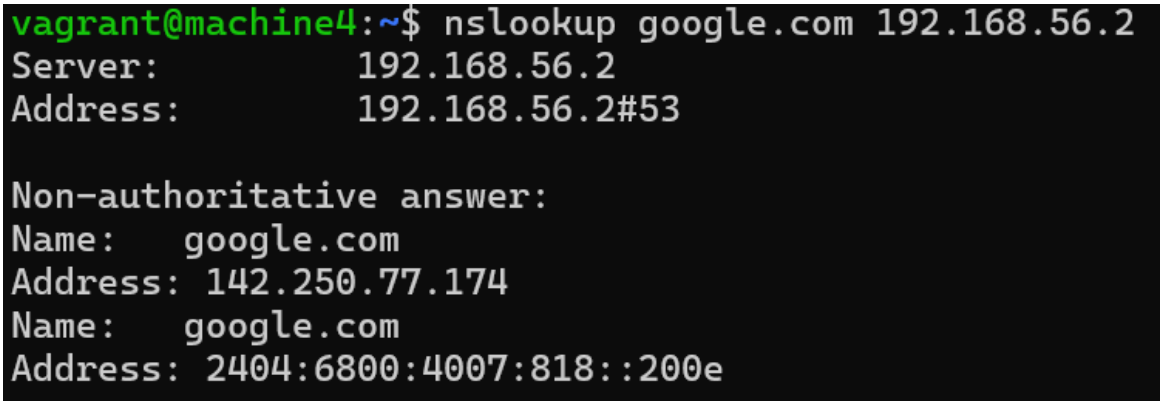

Figure 5.6: Output of nslookup command for External Domains

```
vagrant@machine4:~$ nslookup google.com 192.168.56.2
Server:         192.168.56.2
Address:        192.168.56.2#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.193.110
Name:   google.com
Address: 2404:6800:4007:826::200e
```

Figure 5.7: Output of nslookup command without Forwarders

```
vagrant@machine4:~$ nslookup google.com 192.168.56.2
Server:         192.168.56.2
Address:        192.168.56.2#53

** server can't find google.com: REFUSED
```

Figure 5.8: Output of nslookup command without Recursion

through recursion.

To further verify this, after removing the particular client where resolution is requested from the "trusted" ACL and querying again, we see as shown in figure 5.8 that the record cannot be found. This validates the idea that recursion is happening as expected.

Chapter 6

Conclusions and Future Work

From the setup, testing and results, we come to the conclusion that the authoritative and recursive nameservers have been successfully implemented. The authoritative nameserver gives responses to queries corresponding to records in the zone files defined. The recursive resolver responds with the IP address of the external domains queried by asking each each nameserver in the memory heirarchy until it gets the IP address as discussed previously.

Future work includes adding scalability, privacy, caching and security aspects to this resolver. Scalability features like requests per second may be configured. Performance and other issues have to be monitored and logged. Some of the security features like DNSSEC are provided by BIND and need to be just configured. Deploying a public facing nameserver will require close communication and working with CCC. A container may be allocated where the resolver can be set up. For an IPv4 WAN, NAT will be used whereas for IPv6 can be routed through the L4 router. All of the setup are to the well documented with updates made whenever are made to the server. There should also be a community fro support mainly including CCC and students who worked on this project. With this, a public facing DNS resolver can be set up for NITK.

Bibliography

- A. Affinito, A. Botta and G. Ventre, "Local and Public DNS Resolvers: do you trade off performance against security?," 2022 IFIP Networking Conference (IFIP Networking), Catania, Italy, 2022, pp. 1-9, doi: 10.23919/IFIPNetworking55013.2022.9829756. keywords: Knowledge engineering;Codes;Protocols;Phishing;Malware;Internet;Behavioral sciences;DNS resolver;Domain Name System;DNS over HTTPs;public DNS resolvers;local DNS resolvers

- K. Jerabek, O. Rysavy and I. Burgetova, "Analysis of Well-Known DNS over HTTPS Resolvers," 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2023, pp. 0516-0524, doi: 10.1109/CCWC57344.2023.10099347. keywords: Privacy;Protocols;Conferences;Internet;Encryption;Blocklists;Security;DNS over HTTPS;DoH;DNS security;Security;Privacy;Measurement

- Michael Dooley; Timothy Rooney, "Introduction to the Domain Name System (DNS)," in DNS Security Management , IEEE, 2017, pp.17-29, doi: 10.1002/9781119328292.ch2. keywords: Servers;Internet;Domain Name System;Browsers;World Wide Web;TCPIP

- J. Park, A. Khormali, M. Mohaisen and A. Mohaisen, "Where Are You Taking Me? Behavioral Analysis of Open DNS Resolvers," 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, 2019, pp. 493-504, doi: 10.1109/DSN.2019.00057. keywords: Servers;IP networks;Internet;Computer crime;Standards;Reliability;open resolver;DNS;measurement;behavioral analysis

- Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. 2015. Going Wild: Large-Scale Classification of Open DNS Resolvers. In Proceedings of the 2015 Internet Measurement Conference (IMC '15). Association for Computing Machinery, New York, NY, USA, 355–368. https://doi.org/10.1145/2815675.2815683

- Yingdi Yu, Duane Wessels, Matt Larson, and Lixia Zhang. 2012. Authority server selection in DNS caching resolvers. SIGCOMM Comput. Commun. Rev. 42, 2 (April 2012), 80–86. https://doi.org/10.1145/2185376.2185387

- Google. 2018. "Introduction to Google Public DNS" Performance Benefits. https://developers.google.com/speed/public-dns/docs/performance