Overview

The goal of this project is to use Python to create a basic DNS server. For a limited number of hostnames, this service was created to handle A and CNAME record requests. It illustrates the basic ideas of UDP socket-based server-client communication and DNS resolution.

A dictionary of DNS records that maps hostnames to IP addresses or CNAME aliases is kept up to date by the server. In order to extract the hostname and query type, it evaluates incoming DNS queries. The server creates a response with the matching IP address or CNAME alias based on this data. Additionally, the server gracefully handles problems, including the unavailability of a requested hostname.

There are two parts to the project: the client implementation and the server implementation. The server implementation receives requests, executes queries, produces answers, and manages failures. The user can input a hostname into the client implementation, which then queries the server for DNS information. After that, it gets the server's response and displays it.

This project offers practical experience with UDP communication, the DNS protocol, the server-client concept, and error management techniques. It can be made much better by adding recursion, caching, extending record types, and strengthening security.

Moreover, anyone interested in network communication and DNS resolution can learn a lot from this DNS server project. It uses UDP sockets to illustrate the fundamentals of DNS operation and server-client communication. Understanding the project's execution and possible enhancements can help people comprehend DNS and network programming ideas on a deeper level.

Server development

Server.py is where the server code is implemented. It makes use of the socket module to open a UDP socket on port 53 and wait for requests. A dictionary of records that maps hostnames to IP addresses or CNAME aliases is kept up to date by the server.

The hostname and query type (A or CNAME) are extracted from a request by the server after it has been parsed. The server builds a response with the relevant IP address or CNAME alias if the hostname is found in the records dictionary. It returns an error message otherwise. The client is then provided the response back.

Client development

Client.py is where the client code is implemented. It queries the server for DNS information and lets the user enter a hostname. After receiving it from the server, the client shows the user the response.

Summary and Reflection

Creating this basic DNS server has been a valuable learning experience that has improved my knowledge of DNS resolution and network communication. The project exposed to a number of important ideas, such as:

- DNS protocol: - I developed an in-depth knowledge of the structure of DNS queries and responses, including A, CNAME, MX, PTR, and TXT record types. With this expertise, I was able to efficiently parse incoming requests, obtain relevant information, and create accurate responses.
- UDP sockets: - I gained experience sending and receiving data across networks using UDP connections, which helped me create effective server-client communication. This involved being aware of the ideas behind socket binding, port numbers, and datagrams.
- Server-Client communication: - I established together a basic server-client architecture in which requests are sent by the client and responses are sent by the server. This required building the server with the ability to receive requests, process them, and provide relevant data.
- Error management: - I gained knowledge on how to accept mistakes politely, as when a requested hostname cannot be located on the server. This involved putting error-checking systems in place and giving clients informative error messages.

Moreover, while coming across i gain a huge knowledge and understanding on A and CNAME records,

1. A Record (Address Record)
    - A hostname and an IPv4 address are mapped by an A record. It is in charge of translating human-readable domain names into the numerical IP addresses that computers use to communicate over the internet. It is the most basic record type in DNS. An A record consists of 2 parts,
        - [ ] Hostname: - the name of the domain that is being translated into an IP address.
        - [ ] IPv4 Address: - The 32-bit numerical address that corresponds to the hostname.

Example: - In my code, the records dictionary contains an A record for hacker.com that maps it to the IP address 192.168.1.1. This means that when a client queries for the IP address of hacker.com, the server will respond with 192.168.1.1.

2. CNAME Record (Canonical Name Record)
    - Another hostname is represented through an alias in a CNAME record. In order to manage DNS records or conceal the real server location, it offers a

mechanism to reroute a domain name to another domain name, which may be more practical. CNAME also consists of 2 parts,

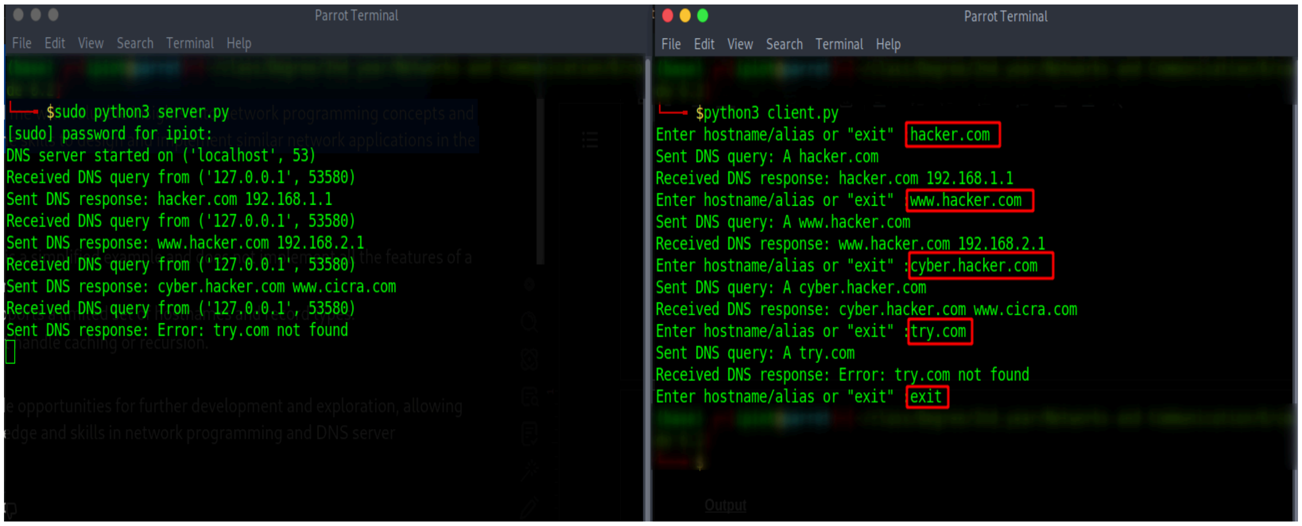
- ☐ Alias Hostname: - The domain name that is being used as an alias.
- ☐ Canonical Hostname: - The actual domain name that the alias points to.

Example: - In my code, the records dictionary contains a CNAME record for www.hacker.com that points to hacker.com. This means that when a client queries for the IP address of www.hacker.com, the server will respond with the IP address of hacker.com, which is 192.168.1.1.

Overall, this project has significantly enhanced my understanding of DNS servers' operations and client interactions in order to resolve hostnames to IP addresses. I've gained important understanding of network programming ideas from the experience, and I now have the know-how to create and carry out related network applications in the future.

NOTE: - The implementation of this project does not include all the features of a real world DNS server, this only supports limited hostnames and record types. Moreover this server does not handle caching or recursion, for it to be done we need to develop it further more. I have done it for the assignment requirement only.

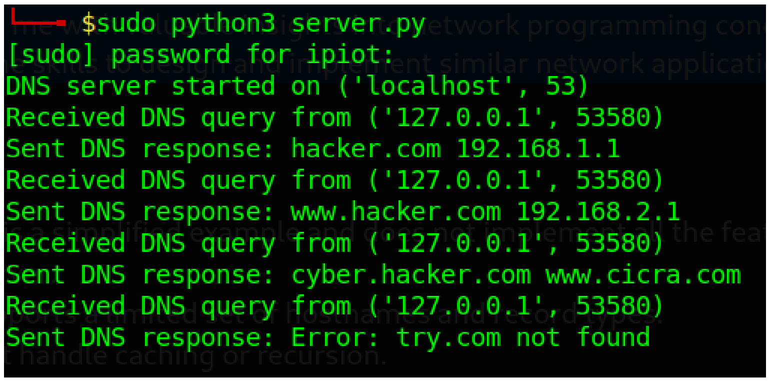
Output



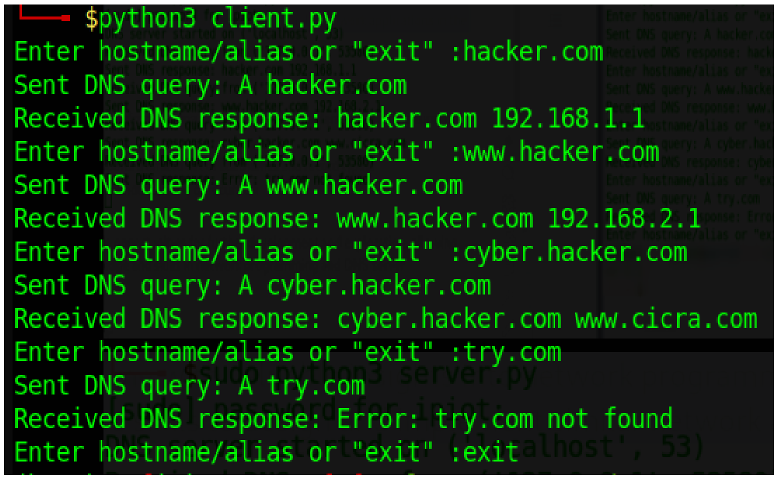
Parrot Terminal
File Edit View Search Terminal Help
Parrot Terminal
File Edit View Search Terminal Help

```
└──╼ $sudo python3 server.py
[sudo] password for ipiot:
DNS server started on ('localhost', 53)
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: hacker.com 192.168.1.1
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: www.hacker.com 192.168.2.1
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: cyber.hacker.com www.cicra.com
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: Error: try.com not found
```

```
└──╼ $python3 client.py
Enter hostname/alias or "exit" :hacker.com
Sent DNS query: A hacker.com
Received DNS response: hacker.com 192.168.1.1
Enter hostname/alias or "exit" :www.hacker.com
Sent DNS query: A www.hacker.com
Received DNS response: www.hacker.com 192.168.2.1
Enter hostname/alias or "exit" :cyber.hacker.com
Sent DNS query: A cyber.hacker.com
Received DNS response: cyber.hacker.com www.cicra.com
Enter hostname/alias or "exit" :try.com
Sent DNS query: A try.com
Received DNS response: Error: try.com not found
Enter hostname/alias or "exit" :exit
```



```
└──╼ $sudo python3 server.py
[sudo] password for ipiot:
DNS server started on ('localhost', 53)
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: hacker.com 192.168.1.1
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: www.hacker.com 192.168.2.1
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: cyber.hacker.com www.cicra.com
Received DNS query from ('127.0.0.1', 53580)
Sent DNS response: Error: try.com not found
```



```
└──╼ $python3 client.py
Enter hostname/alias or "exit" :hacker.com
Sent DNS query: A hacker.com
Received DNS response: hacker.com 192.168.1.1
Enter hostname/alias or "exit" :www.hacker.com
Sent DNS query: A www.hacker.com
Received DNS response: www.hacker.com 192.168.2.1
Enter hostname/alias or "exit" :cyber.hacker.com
Sent DNS query: A cyber.hacker.com
Received DNS response: cyber.hacker.com www.cicra.com
Enter hostname/alias or "exit" :try.com
Sent DNS query: A try.com
Received DNS response: Error: try.com not found
Enter hostname/alias or "exit" :exit
```

In this demonstration I have shown how the client requests for the IP address of hacker.com the server replies with 192.168.1.1, as I have done for two more CNAME records. And also I have shown an error handling mechanism as when I enter “try.com” the request is sent to the server and gets a response as the domain is not found as the A record will check for its mapped IP address and when it's not there it gives back a response as not found. Also to leave out i have just entered “exit”. This is a simple demonstration of my DNS server project.