The implementation of a custom Domain Name Server

Dariya Vakhitova, Arslanov Shamil
Innopolis University, 2021.

Abstract

We describe the idea and implementation of a custom Domain Name System, using C socket programming for network communication and SQLite3 database for data storage. The code of our implementation is available publicly at the GitHub repository https://github.com/homomorfism/dns_server.

1. Introduction

The Domain Name System provides an indispensable substrate for Internet applications and services by linking human-friendly names with machine-readable addresses. Being a deceptively simple protocol based on the client-server model, the DNS basically forwards users queries to authoritative servers, requesting for the translation from name to address or vice visa, which then returns the response.

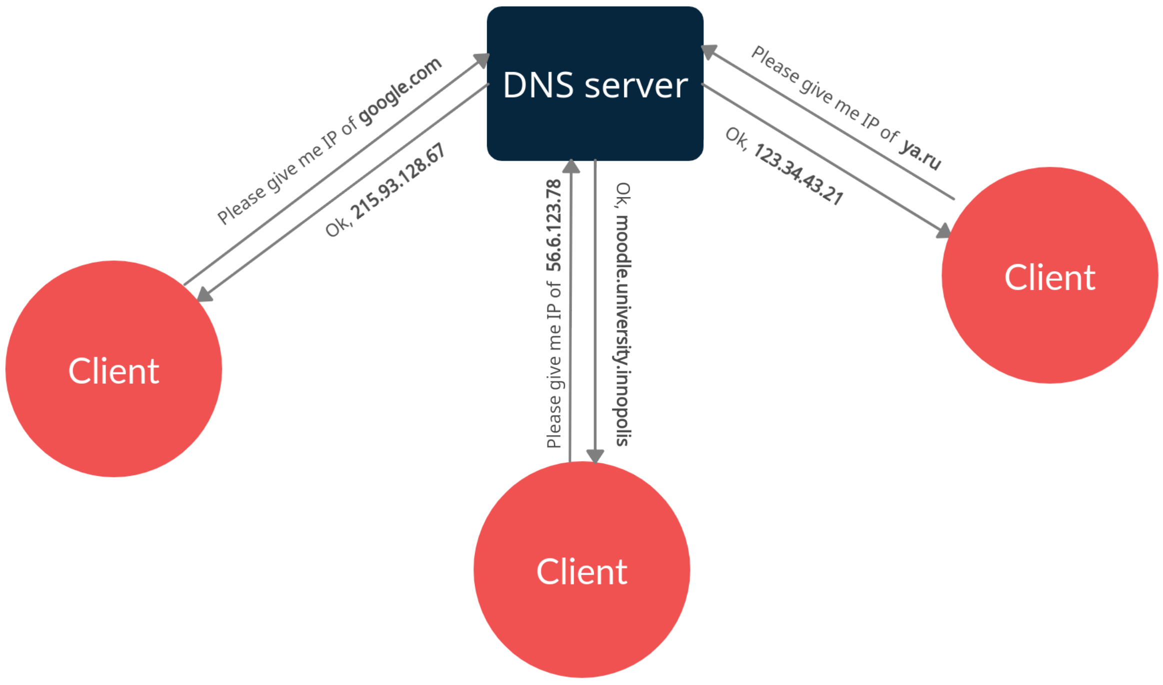

DNS server
Please give me IP of google.com
Ok, 215.93.128.67
Please give me IP of ya.ru
Ok, 123.34.43.21
Please give me IP of 56.6.123.78
Ok, moodle.university.innopolis
Client
Client
Client

Figure 1. The high-level view of the principle of working with a DNS server.

DNS protocol plays a significant role in providing fast and reliable access to the Internet: conversion of the IP addresses and domain names should apply fast and should be constantly accessible to clients.

We next describe our implementation experience of a custom DNS by using C socket programming for network communication and SQLite 3 database in the scope of computer network course at the Innopolis University.

2. Implementation of a Custom DNS Server

We below explain the key implementation aspects of the Domain Name System server.

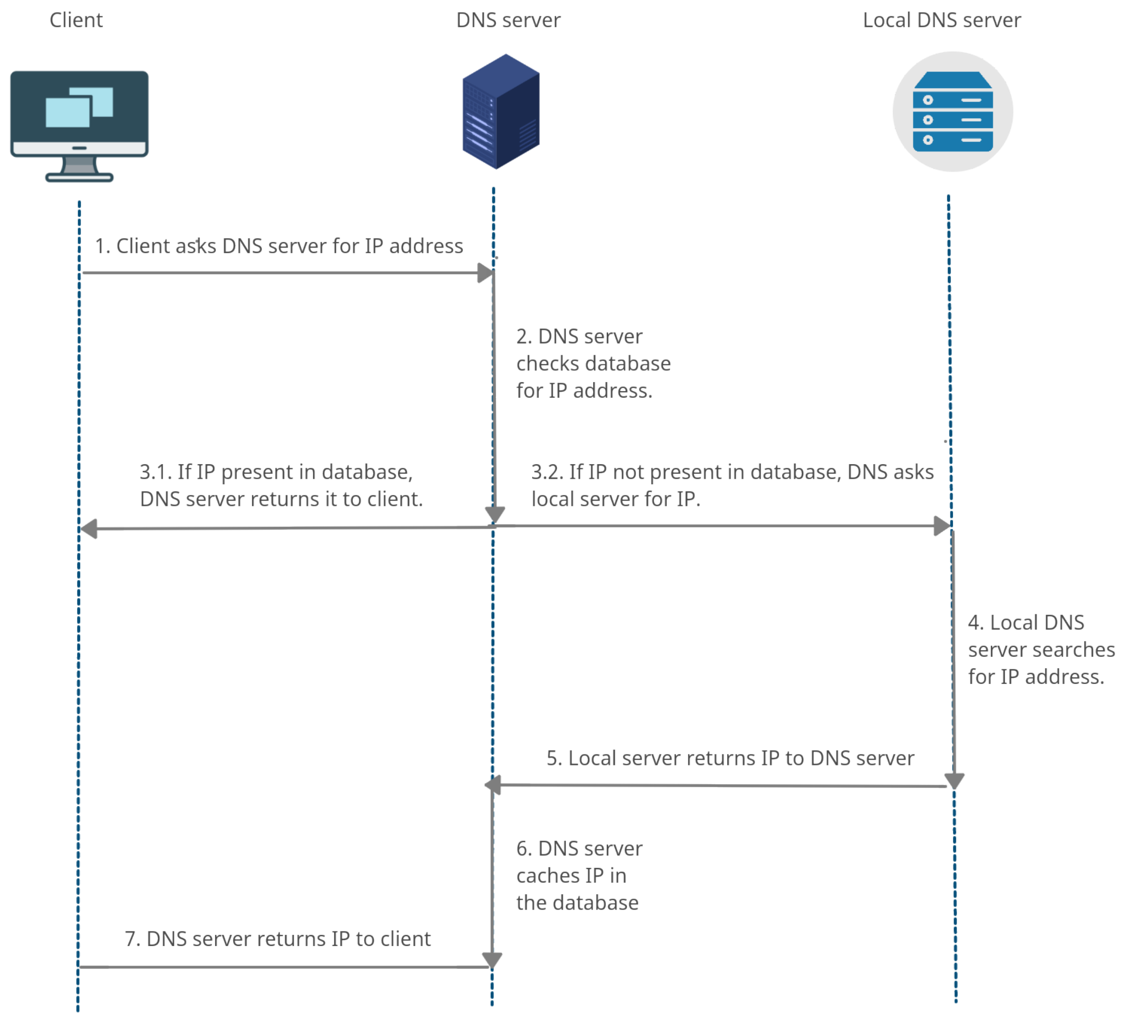
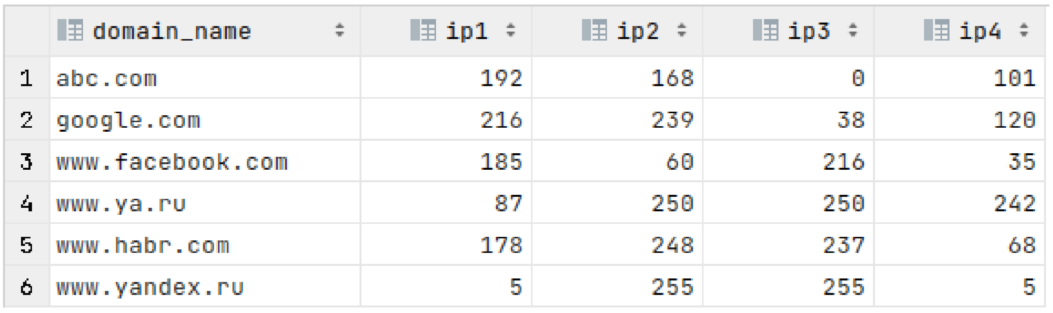

Client
DNS server
Local DNS server
1. Client asks DNS server for IP address
2. DNS server checks database for IP address.
3.1. If IP present in database, DNS server returns it to client.
3.2. If IP not present in database, DNS asks local server for IP.
4. Local DNS server searches for IP address.
5. Local server returns IP to DNS server
6. DNS server caches IP in the database
7. DNS server returns IP to client

Figure 2. Principle of working of our Custom DNS server: a client, using **dig - CLI tool for communicating with DNS servers**, sends to the DNS server a request to convert a domain name into an IP address. A DNS server connects to a local DNS server and retrieves the IP that is sent to the client.

We implement a Domain Name System server that stores domain names and IP addresses associated with them. The project consists of an authoritative DNS server and an API for connecting to a local server.

The authoritative DNS server has a built-in SQLite 3 database, where the corresponding mapping of hostname to IP address is stored. When the client requests the IP, the authoritative server first checks if the requested mapping of hostname to IP is located in the database. In case IP is stored in a database the server retrieves it and responds to the user. In case if IP is not saved in the database the authoritative server makes a call to the local DNS server.

The address of the local DNS server is obtained from the configuration file of the Authoritative server. When the authoritative server fails to retrieve an IP from the database, it makes a call to the local DNS server. After obtaining information the local server returns it to the authoritative server, which saves information to the SQLite 3 database and only then returns it to the user.



	domain_name	ip1	ip2	ip3	ip4
1	abc.com	192	168	0	101
2	google.com	216	239	38	120
3	www.facebook.com	185	60	216	35
4	www.ya.ru	87	250	250	242
5	www.habr.com	178	248	237	68
6	www.yandex.ru	5	255	255	5

Figure 3. SQLite 3 database system used in the project.

3. Static Program Analysis

We next provide static analysis of the application. We measure the execution time of the program on three different cases using *time* command from *time.h* C-library.

- The first case - IP address is stored in the database. Time is less than 8 msec.
- Second case - IP not in the database and should be retrieved from a local Domain Name System server. Time is approximately 200 msec.
- The third case - IP not in the database nor in the local Domain Name System server. Time is approximately 20 msec.

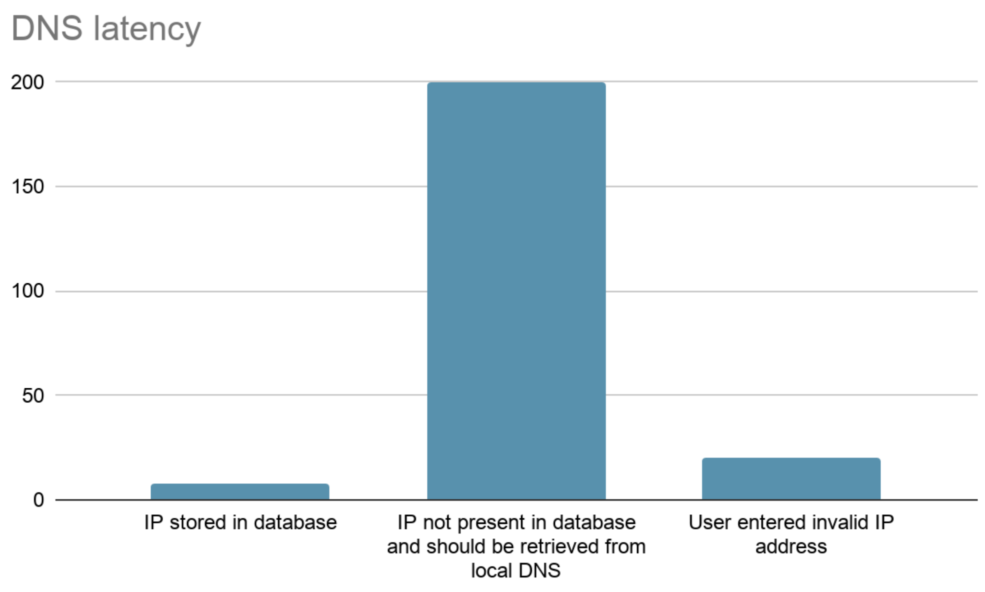


Figure 4. Measuring response time of various use cases.

4. Program Testing

We next show how to run and test the application.

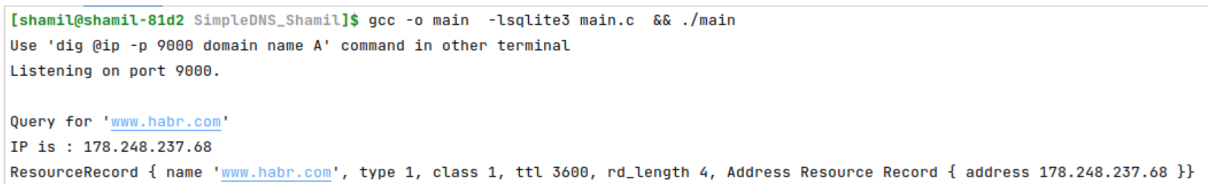
The commands provided are suitable for Linux based operating systems only. It is important to have a C compiler to run the application.

Running DNS server :

```
gcc -o main -lsqlite3 main.c  && ./main
```

Accessing DNS server via terminal:

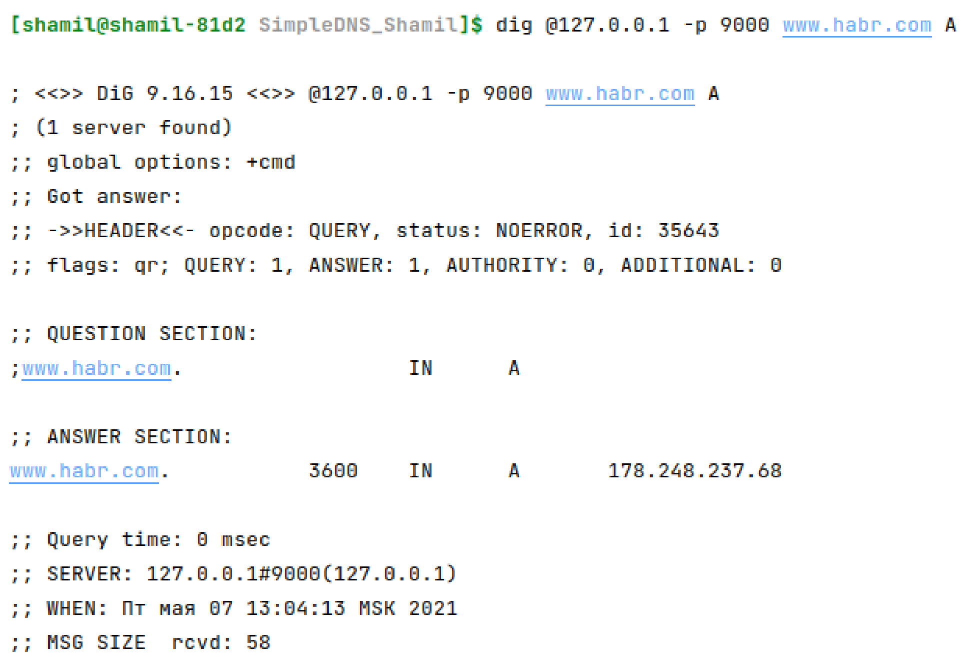
```
dig @<IPv4 address> -p <port number> <domain name> A
```



```
[shamil@shamil-81d2 SimpleDNS_Shamil]$ gcc -o main  -lsqlite3 main.c  && ./main
Use 'dig @ip -p 9000 domain name A' command in other terminal
Listening on port 9000.

Query for 'www.habr.com'
IP is : 178.248.237.68
ResourceRecord { name 'www.habr.com', type 1, class 1, ttl 3600, rd_length 4, Address Resource Record { address 178.248.237.68 }}
```

Figure 4. Logs from the DNS server, obtained from resolving domain name (www.habr.com) to IP address (178.248.237.68).



```
[shamil@shamil-81d2 SimpleDNS_Shamil]$ dig @127.0.0.1 -p 9000 www.habr.com A

; <<>> DiG 9.16.15 <<>> @127.0.0.1 -p 9000 www.habr.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35643
;; flags: qr; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.habr.com.                  IN      A

;; ANSWER SECTION:
www.habr.com.           3600    IN      A       178.248.237.68

;; Query time: 0 msec
;; SERVER: 127.0.0.1#9000(127.0.0.1)
;; WHEN: Пт мая 07 13:04:13 MSK 2021
;; MSG SIZE  rcvd: 58
```

Figure 5. DNS packet, received from the server by client, containing resolved IP address (178.248.237.68) of (www.habr.com) domain name.

5. Limitations of the Project

We next describe the limitations of the project.

For the simplicity of the project, we limit the functionality of the server. Firstly, the server is not constantly accessible to clients - it could process only one query at a time and then exits the program. Secondly, the server does not support concurrent fulfilling queries, so multiple users can not access it. Thirdly, the server only supports A type of queries. Moreover, the server can run only on Linux-based systems due to special libraries which are working only on that family of the operating system.

6. Conclusion

We proposed the implementation of a DNS server that handles request IP and domain name conversion among clients. We discovered a decrease of latency of DNS resolving using a built-in SQLite 3 database that caches recently retrieved IP addresses. During the project, we deeply studied the main principles of the DNS resolution process, C socket programming API and best practices of working with database systems. However, more knowledge is needed to optimize our application.