

DNSRelay - Group 24

Group Members: Tiantian Li, He Zhu, Tianyi Yang

Overview

Brief Requirements

Implement a DNS relay that:

- Receives DNS queries from DNS clients and forwards them to a given DNS server.
- Receives DNS responses from the DNS server and forwards them to the clients.

There are 3 different cases we need to handle:

- For domain name included in the local database (e.g., hosts.txt), it sends back the corresponding IP addresses.
- If found, for IP address 0.0.0.0, it sends back "no such name" (reply code=0011).
- For domain name not included in the database, it forwards the query to the DNS server.

Target

One of the targets is to gain a deeper understanding of how the Domain Name System (DNS) works and, specifically, to learn about the process of resolving domain names to IP addresses, which is a fundamental aspect of how the Internet functions.

By implementing a DNS relay in Rust, we will have the opportunity to learn how to work with Rust's syntax and data types, as well as its concurrency model. Rust has a strong focus on safe and efficient concurrency, which makes it a great choice for building networked applications like a DNS relay.

Requirements Analysis

Development Environment

- Operating system: Arch Linux
- Programming language: Rust 1.70.0

Detailed Requirements

On startup, the program should read environmental variables, user arguments and the local hosts file. It also opens two `UdpSockets`: one for communicating with clients, one for communicating with upstream DNS server.

Upon receiving queries from client, the program parses the packet and extract useful information for further process. The hosts file is looked up for local answer construction and blacklist blocking.

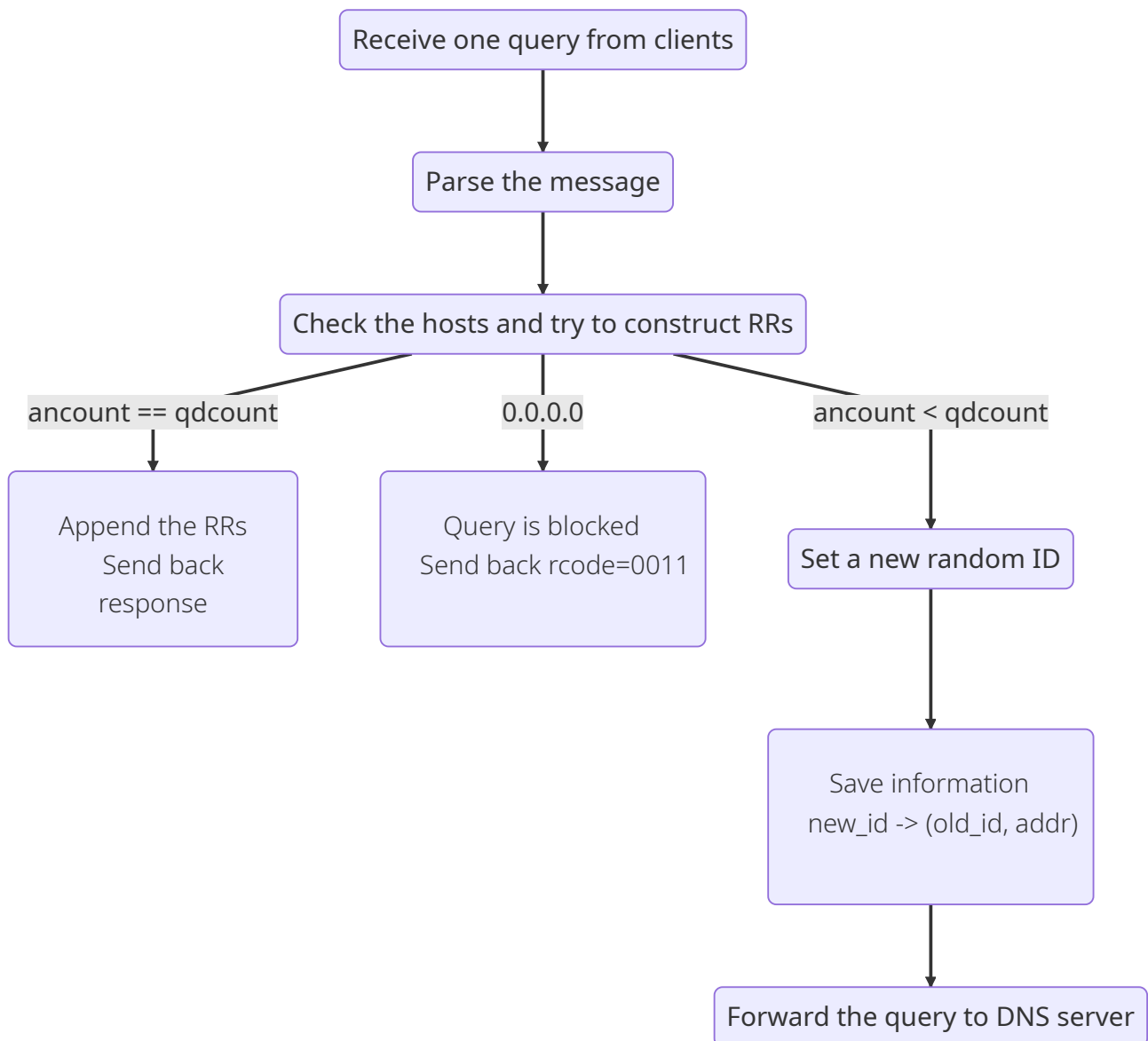
If all questions in the query can be processed without consulting the upstream DNS server, a reply consisting of one or multiple answers is constructed and sent to the clients. Otherwise, the query packet is forwarded to the upstream DNS server.

System Design

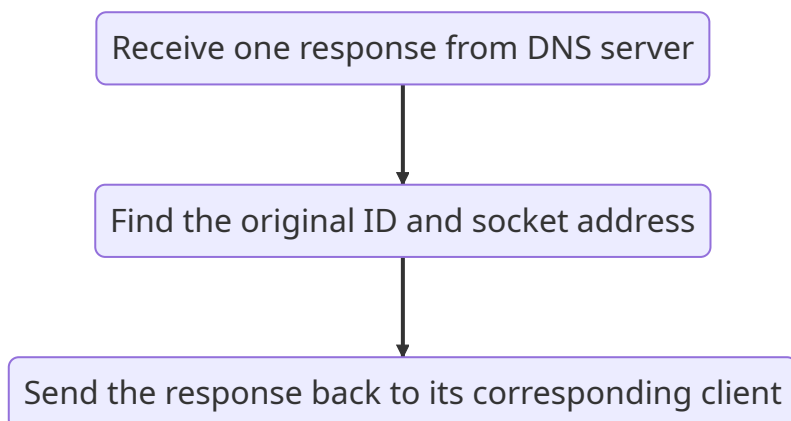
Generally speaking, there are two loops running **asynchronously**, namely 'forward' and 'reply'. The following diagrams show the their workflows.

Flow Chart

Workflow - Forward



Workflow - Reply



Module Decomposition

The application consists of three parts serving different functionalities:

- `main.rs` – the entry of the program
- `lib.rs` – the business logic, including the two workflows mentioned above configuration reading
- `packet.rs` – the homemade wrappers for DNS packets

Data Structures

Struct `Message` comprises a header, a question, and an answer struct. All three types of struct consist of a mutable reference to a byte buffer and the length of the buffer. Methods such as `get_id()` and `add_entries()` are implemented to manipulate the underlying data.

```
pub struct Message<'a> {  
    pub header: Header<'a>,  
    pub question: Question<'a>,  
    pub answer: Answer<'a>,  
}
```

Struct `QuestionEntry` is comprised of an offset, a `qname`, a `qtype` and a `qclass`. It is used to represent the parsed version of entries in the question section.

The offset points to the starting byte of the `qname`. The `qname` is the query string. The `qtype` field is used to specify the type of resource record being requested. Common types include A records (which map domain names to IP addresses), MX records (which specify the mail server for a domain), and NS records (which specify the authoritative name server for a domain). The `qclass` field is used to specify the class of the resource record being requested. This is typically set to IN, which indicates that the record is part of the Internet class.

```
pub struct QuestionEntry {  
    pub offset: usize,  
    pub qname: String,  
    pub qtype: u16,  
    pub qclass: u16,  
}
```

Struct `ResourceRecord` contains the information needed to construct the reply packet. It includes a name, a `rtype`, a `rclass`, a `ttd`, a `rdlength`, and a `rdata`. The name is stored as a pointer for message compression.

```
pub struct ResourceRecord {
    pub name: u16,
    pub rtype: u16,
    pub rclass: u16,
    pub ttl: u32,
    pub rdlength: u16,
    pub rdata: RData,
}

pub enum RData {
    V4([u8; 4]),
    V6([u8; 16]),
}
```

Testings and Results

Sample hosts file:

```
# ./hosts.txt
0.0.0.0 www.baidu.com www.zhihu.com www.qq.com
211.68.69.240 www.bupt.edu.cn
```

1. Blacklist

As shown in the `hosts.txt`, `www.baidu.com` is blocked with an address of `0.0.0.0`.

The program responded with a `NXDOMAIN`, indicating that the domain did not exist.

```
arcohol@arch-laptop ~/p/mini-dns-relay (master)> sudo UPSTREAM_ADDR=223.5.5.53 ./target/release/min
i-dns-relay -vv
2023-07-02T15:53:35.452384Z INFO mini_dns_relay: config: Config { local_addr: "127.0.0.1:53", remote
_addr: "0.0.0.0:10053", upstream_addr: "223.5.5.53", hosts_path: "hosts.txt" }
2023-07-02T15:53:35.452516Z INFO mini_dns_relay: local socket is listening on 127.0.0.1:53
2023-07-02T15:53:35.452550Z INFO mini_dns_relay: remote socket is listening on 0.0.0.0:10053
2023-07-02T15:53:35.452635Z DEBUG mini_dns_relay: hosts: {"www.baidu.com": 0.0.0.0, "www.zhihu.com":
0.0.0.0, "www.qq.com": 0.0.0.0, "www.bupt.edu.cn": 211.68.69.240}
2023-07-02T15:53:38.967652Z INFO mini_dns_relay: (5706) query received from 127.0.0.1:53661
2023-07-02T15:53:38.967695Z DEBUG mini_dns_relay: (5706) questions parsed: [QuestionEntry { offset: 1
2, qname: "www.baidu.com", qtype: 1, qclass: 1 }]
2023-07-02T15:53:38.967712Z INFO mini_dns_relay: (5706) query is blocked, sending response back to 1
27.0.0.1:53661
█
```

```
arcohol@arch-laptop ~/p/mini-dns-relay (master)> nslookup www.baidu.c
om 127.0.0.1
Server:      127.0.0.1
Address:     127.0.0.1#53

** server can't find www.baidu.com: NXDOMAIN

arcohol@arch-laptop ~/p/mini-dns-relay (master) [1]> █
```

2. Local Record Matching

`www.bupt.edu.cn` exists as an entry in `hosts.txt`, so the relay server successfully constructed a local resource record and returned the recorded A answer `211.68.69.240`.

The AAAA query which cannot be processed locally is forwarded to the upstream.

```
arcohol@arch-laptop ~/p/mini-dns-relay (master)> sudo UPSTREAM_ADDR=223.5.5.53 ./target/release/mini-dns-relay -vv
2023-07-02T15:51:26.686072Z INFO mini_dns_relay: config: Config { local_addr: "127.0.0.1:53", remote_addr: "0.0.0.0:10053", upstream_addr: "223.5.5.53", hosts_path: "hosts.txt" }
2023-07-02T15:51:26.686155Z INFO mini_dns_relay: local socket is listening on 127.0.0.1:53
2023-07-02T15:51:26.686178Z INFO mini_dns_relay: remote socket is listening on 0.0.0.0:10053
2023-07-02T15:51:26.686220Z DEBUG mini_dns_relay: hosts: {"www.qq.com": 0.0.0.0, "www.bupt.edu.cn": 211.68.69.240, "www.zhihu.com": 0.0.0.0, "www.baidu.com": 0.0.0.0}
2023-07-02T15:51:29.552535Z INFO mini_dns_relay: (612e) query received from 127.0.0.1:51394
2023-07-02T15:51:29.552580Z DEBUG mini_dns_relay: (612e) questions parsed: [QuestionEntry { offset: 1, qname: "www.bupt.edu.cn", qtype: 1, qclass: 1 }]
2023-07-02T15:51:29.552596Z DEBUG mini_dns_relay: (612e) local rr created: ResourceRecord { name: c00c, rtype: 1, rclass: 1, ttl: 258, rlength: 4, rdata: V4([d3, 44, 45, f0]) }
2023-07-02T15:51:29.552611Z DEBUG mini_dns_relay: (612e) constructed a total of 1 local rr(s)
2023-07-02T15:51:29.552621Z INFO mini_dns_relay: (612e) query is processed locally, sending response back to 127.0.0.1:51394
2023-07-02T15:51:29.553102Z INFO mini_dns_relay: (a509) query received from 127.0.0.1:47408
2023-07-02T15:51:29.553125Z DEBUG mini_dns_relay: (a509) questions parsed: [QuestionEntry { offset: 1, qname: "www.bupt.edu.cn", qtype: 28, qclass: 1 }]
2023-07-02T15:51:29.553140Z INFO mini_dns_relay: (a509) query cannot be processed locally
2023-07-02T15:51:29.553162Z INFO mini_dns_relay: (a509) new id generated: d2ac
2023-07-02T15:51:29.553171Z INFO mini_dns_relay: (d2ac) query is sending to upstream
2023-07-02T15:51:29.569092Z INFO mini_dns_relay: (d2ac) response received from upstream
2023-07-02T15:51:29.569128Z INFO mini_dns_relay: (d2ac) the original query id is a509, changing back to it
2023-07-02T15:51:29.569136Z INFO mini_dns_relay: (a509) upstream response is sending back to 127.0.0.1:47408
█

arcohol@arch-laptop ~/p/mini-dns-relay (master)> nslookup www.bupt.edu.cn 127.0.0.1
Server:      127.0.0.1
Address:     127.0.0.1#53

Non-authoritative answer:
Name:   www.bupt.edu.cn
Address: 211.68.69.240
www.bupt.edu.cn canonical name = vn46.bupt.edu.cn.
Name:   vn46.bupt.edu.cn
Address: 2001:da8:215:4038::161

arcohol@arch-laptop ~/p/mini-dns-relay (master)> █
```

3. Upstream Forwarding

www.apple.com was not found in the hosts. Therefore, the program forwarded the packet to upstream DNS server. After receiving the reply from the upstream, it forwarded back to the client.

```
arcohol@arch-laptop ~/p/mini-dns-relay (master)> sudo UPSTREAM_ADDR=223.5.5.53 ./target/release/mini-dns-relay -vv
2023-07-02T15:59:28.330892Z INFO mini_dns_relay: config: Config { local_addr: "127.0.0.1:53", remote_addr: "0.0.0.0:10053", upstream_addr: "223.5.5.53", hosts_path: "hosts.txt" }
2023-07-02T15:59:28.330195Z INFO mini_dns_relay: local socket is listening on 127.0.0.1:53
2023-07-02T15:59:28.330215Z INFO mini_dns_relay: remote socket is listening on 0.0.0.0:10053
2023-07-02T15:59:28.330264Z DEBUG mini_dns_relay: hosts: {"www.zhihu.com": 0.0.0.0, "www.qq.com": 0.0.0.0, "www.baidu.com": 0.0.0.0, "www.bupt.edu.cn": 211.68.69.240}
2023-07-02T15:59:42.596933Z INFO mini_dns_relay: (37c3) query received from 127.0.0.1:58121
2023-07-02T15:59:42.596974Z DEBUG mini_dns_relay: (37c3) questions parsed: [QuestionEntry { offset: 1, qname: "www.apple.com", qtype: 1, qclass: 1 }]
2023-07-02T15:59:42.596987Z INFO mini_dns_relay: (37c3) query cannot be processed locally
2023-07-02T15:59:42.597016Z INFO mini_dns_relay: (37c3) new id generated: db61
2023-07-02T15:59:42.597026Z INFO mini_dns_relay: (db61) query is sending to upstream
2023-07-02T15:59:42.611038Z INFO mini_dns_relay: (db61) response received from upstream
2023-07-02T15:59:42.611075Z INFO mini_dns_relay: (db61) the original query id is 37c3, changing back to it
2023-07-02T15:59:42.611084Z INFO mini_dns_relay: (37c3) upstream response is sending back to 127.0.0.1:58121
2023-07-02T15:59:42.611775Z INFO mini_dns_relay: (15d3) query received from 127.0.0.1:35866
2023-07-02T15:59:42.611795Z DEBUG mini_dns_relay: (15d3) questions parsed: [QuestionEntry { offset: 1, qname: "e6858.e19.s.tl88.net", qtype: 28, qclass: 1 }]
2023-07-02T15:59:42.611812Z INFO mini_dns_relay: (15d3) query cannot be processed locally
2023-07-02T15:59:42.611822Z INFO mini_dns_relay: (15d3) new id generated: 52b4
2023-07-02T15:59:42.611836Z INFO mini_dns_relay: (52b4) query is sending to upstream
2023-07-02T15:59:42.792648Z INFO mini_dns_relay: (52b4) response received from upstream
2023-07-02T15:59:42.792699Z INFO mini_dns_relay: (52b4) the original query id is 15d3, changing back to it
2023-07-02T15:59:42.792708Z INFO mini_dns_relay: (15d3) upstream response is sending back to 127.0.0.1:35866
█

arcohol@arch-laptop ~/p/mini-dns-relay (master)> nslookup www.apple.com 127.0.0.1
Server:      127.0.0.1
Address:     127.0.0.1#53

Non-authoritative answer:
www.apple.com canonical name = www.apple.com.edgekey.net.
www.apple.com.edgekey.net canonical name = www.apple.com.edgekey.net.globalredir.akadns.net.
www.apple.com.edgekey.net.globalredir.akadns.net canonical name = e6858.e19.s.tl88.net.
Name:   e6858.e19.s.tl88.net
Address: 223.111.101.29

arcohol@arch-laptop ~/p/mini-dns-relay (master)> █
```

Conclusion and Future Improvements

The implementation of a DNS relay is a challenging and rewarding project that provides a valuable learning experience in network programming. Through this project, we have gained a deeper understanding of the Domain Name System (DNS) and how it facilitates internet communication by resolving domain names to IP addresses.

Our implementation of the DNS relay in Rust allowed us to learn and improve our skills in this modern systems programming language. Rust's features and focus on safe and efficient concurrency made it an excellent choice for building a networked application like a DNS relay.

Throughout the implementation process, we faced a number of challenges, including working with DNS queries and responses. However, through a methodical approach to problem-solving and careful consideration of system design, we were able to overcome these challenges and produce a functional and reliable DNS relay server.

There are several possible future improvements that can be employed in this system. For example, a fast and reliable caching system can be implemented so that the network consumption for upstream link will be greatly reduced. It can be very tricky as this involves cache time design, and possibly recursive searching. We decided to not include a cache in this system because we think a unreliable cache is pretty much redundant and may cause a significant drop of performance.

Overall, this project has provided us with valuable experience in network programming, Rust development, and system design.