BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

NETWORK APPLICATIONS AND NETWORK ADMINISTRATION

Semester project

DNS resolver

October 20, 2023 Vadim Goncearenco (xgonce00@stud.fit.vutbr.cz)

Contents

1 Theory

1.1 Domain Name System

DNS (Domain Name System) is a hierarchical naming system that maps domain names to IP addresses. It is used to translate human-readable domain names, such as www.example.com, into IP addresses, such as 192.0.2.1, which are used to identify and locate computer services and devices on the internet.[2]

A DNS resolver is a program that queries DNS servers to resolve domain names into IP addresses. When a user enters a domain name into a web browser, the browser sends a DNS query to a DNS resolver, which then sends a query to a DNS server to obtain the IP address associated with the domain name.

The DNS resolver program typically uses the following steps to resolve a domain name:

- The resolver sends a query to a DNS server, asking for the IP address associated with the domain name.
- If the DNS server has the IP address in its cache, it returns the IP address to the resolver.
- If the DNS server does not have the IP address in its cache, it sends a query to another DNS server higher up in the hierarchy.
- This process continues until the IP address is found, or until the query reaches the root DNS server, which returns an error if the domain name is not valid.
- Once the DNS resolver program has obtained the IP address associated with the domain name, it can use this information to establish a connection to the web server associated with the domain name, and retrieve the web page or other content requested by the user.

DNS queries are used to obtain information about domain names, such as IP addresses, mail server information, and other resource records. A DNS query consists of a header and a question section, which contains the domain name being queried and the type of information being requested.

When a DNS query is sent, it is first sent to a local DNS resolver, which checks its cache to see if it has the requested information. If the information is not in the cache, the resolver sends the query to a DNS server.

The DNS server then checks its cache to see if it has the requested information. If the information is not in the cache, the server sends the query to another DNS server higher up in the hierarchy. This process continues until the information is found, or until the query reaches the root DNS server, which returns an error if the domain name is not valid.

Once the DNS server has obtained the requested information, it sends a response back to the DNS resolver, which then sends the information back to the client that made the original query.

DNS queries can be recursive or iterative. In a recursive query, the DNS server is responsible for obtaining the requested information and returning it to the client. In an iterative query, the DNS server returns the best information it has available, and the client is responsible for sending additional queries to obtain more information if needed.

DNS queries can also be performed over UDP or TCP. UDP is used for small queries, while TCP is used for larger queries that require more data to be transmitted.

This section was adopted from `RFC 1035` [2].

1.2 DNS packet format

A DNS packet consists of a header and four sections: question, answer, authority, and additional. The header contains information about the packet, such as the ID, flags, and number of records in each section. The question section contains the domain name being queried and the type of information being requested. The answer section contains the information requested by the query. The authority section contains information about the DNS server that provided the information in the answer section. The additional section contains additional information about the domain name being queried.

The header contains the following fields:

- ID - A 16-bit identifier assigned by the program that generates any kind of query. This identifier is copied in the corresponding reply and can be used by the requester to match up replies to outstanding queries.
- QR - A one bit field that specifies whether this message is a query (0), or a response (1).
- Opcode - A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response.
- AA - Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.
- TC - TrunCation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.
- RD - Recursion Desired - this bit may be set in a query and is copied into the response. If RD is set, it directs the name server to pursue the query recursively.
- RA - Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.
- Z - Reserved for future use. Must be zero in all queries and responses.
- RCODE - Response code - this 4 bit field is set as part of responses. The values have the following interpretation:
  - 0 - No error condition
  - 1

1.3 DNS resolver

2 Application design

The `dns` program is implemented in *C programming language* using *c99* language standard. For network programming the program utilizes *BSD socket libraries*. The following network libraries are used:

- `arpa/inet.h` - Contains functions for converting between network and host byte order.
- `netinet/in.h` - Contains constants and structures needed for internet domain addresses.
- `sys/types.h` - Contains definitions of data types used in system calls.
- `sys/socket.h` - Contains definitions of socket data types.
- `netdb.h` - Contains definitions of network database operations.

2.1 Project files

Project directory contains the following files:

- `dns.c` - Main file. Contains the `main` function and signal handler functions.
- `base.h` - Contains common header includes, macros and constants.
- `args.c` - Contains functions for parsing command line arguments.
- `args.h` - Header file for `args.c`.

- `dns_packet.c` - Contains functions for creating and parsing DNS packets.
- `dns_packet.h` - Header file for `dns_packet.c`.
- `Makefile` - Makefile for building the application and running the tests.
- `README.md` - Contains information about the application.
- `test/test.py` - Python script for running the tests.
- `test/test_cases.json` - Contains test cases for the `test.py` script.
- `doc/manual.pdf` - This document.

3 Program usage

The following is the usage of the `dns` program:

```
NAME
    dns - DNS resolver

SYNOPSIS
    dns [-r] [-x|-6] -s server [-p port] domain|address
    dns -h

DESCRIPTION
    dns is a simple DNS resolver that can handle both IPv4 and IPv6
    addresses. Additionally, it has the capability to perform
    reverse queries and can query any DNS server. It supports
    recursive queries and can also communicate with DNS servers using
    IPv6 or a non-standard port.

OPTIONS
    -r
        Recursive query. If the DNS server does not have the answer,
        it will recursively query other DNS servers.

    -x
        Reverse query. The address is interpreted as an IPv4 address
        and the PTR record is queried. IPv6 addresses are not supported.

    -6
        Force IPv6. The query is sent to the DNS server using IPv6.

    -s server
        DNS server to query.

    -p port
        Port to use when querying the DNS server. Default is 53.

    -h
        Print help and exit.
```

```
    domain|address
        Domain name to query or IPv4 address to reverse query.

SIMPLE USAGE
    ./dns -r -s dns.google www.github.com
    Authoritative: No, Recursive: Yes, Truncated: No
    Question section (1)
        www.github.com., A, IN
    Answer section (2)
        www.github.com., CNAME, IN, 3600, github.com.
    github.com., A, IN, 60, 140.82.121.3
    Authority section (0)
    Additional section (0)

    ./dns -r -x -s dns.google 140.82.121.3
    Authoritative: No, Recursive: Yes, Truncated: No
    Question section (1)
        3.121.82.140.in-addr.arpa., PTR, IN
    Answer section (1)
        3.121.82.140.in-addr.arpa., PTR, IN, 2004,
    lb-140-82-121-3-fra.github.com.
    Authority section (0)
    Additional section (0)

    ./dns -r -s kazi.fit.vutbr.cz www.fit.vut.cz
    Authoritative: No, Recursive: Yes, Truncated: No
    Question section (1)
        www.fit.vut.cz., A, IN
    Answer section (1)
        www.fit.vut.cz., A, IN, 14400, 147.229.9.26
    Authority section (0)
    Additional section (0)

    ./dns -r -s kazi.fit.vutbr.cz www.github.com
    Authoritative: No, Recursive: Yes, Truncated: No
    Question section (1)
        www.github.com., A, IN
    Answer section (2)
        www.github.com., CNAME, IN, 3600, github.com.
        github.com., A, IN, 60, 140.82.121.3
    Authority section (0)
    Additional section (0)
```

The description of program usage can also be found in the `README.md` file.

4 Testing

Testing is done using the `test.py` script. The script reads test cases from the `test_cases.json` file and runs the program with the given arguments. The output of the program is then compared with the output of `dig` Linux utility program [3].

The script can be run using the following `Makefile` target:

```
make test
```

or by running the Python script directly:

```
usage:
    python3 test.py [-h] [-d] [-i] [-v] input_file

positional arguments:
    input_file          input JSON file

options:
    -h, --help          show this help message and exit
    -d, --debug         run in debug mode
                       (print all queries and responses)
    -i, --ignore-ipv6   ignore ipv6 tests (AAAA queries)
    -v, --ignore-vpn    ignore tests that require VUT FIT network VPN
```

Be aware though, that because the test script compares output of the `dns` program with output of the `dig` command, some tests may occasionally fail because both programs may resolve a domain name to a different address depending on the exact time of execution. For example, the domain name `www.github.com` can be resolved to either `140.82.121.3` or `140.82.121.4` IPv4 address. It may be due to the use of *DNS round-robin* technique, which is used to distribute traffic across multiple servers by returning different IP addresses in response to DNS queries [1].

So to ensure that the program works correctly, it is important to run the tests multiple times.

References

[1] Thomas P. Brisco. *DNS Support for Load Balancing*. Tech. rep. RFC 1794. Apr. 1995. DOI: 10.17487/RFC1794. URL: https://datatracker.ietf.org/doc/rfc1794 (visited on 10/20/2023).

[2] *Domain names - implementation and specification*. Tech. rep. RFC 1035. Nov. 1987. DOI: 10.17487/RFC1035. URL: https://datatracker.ietf.org/doc/rfc1035 (visited on 10/20/2023).

[3] *IBM Documentation*. en-US. online. Mar. 2023. URL: https://www.ibm.com/docs/en/aix/7.1?topic=d-dig-command (visited on 10/20/2023).