Documentation of Project 1 **Monitoring DNS communication** variant for ISA 2024/2025

Name and surname: **Andrii Klymenko**

Login: **xklyme00**

1 Introduciton

DNS is one of the most important services on the Internet. People use it dozens of times every day without even realizing it: when they visit a website (through domain names, like apple.com or eshop.notime), when they send e-mails, etc. In order to understand how this service works, what are its features, I decided to implement a program that will monitor DNS communication and process it - DNS monitor.

2 Theory

In this chapter there is presented the basic information that was necessary to study before implementing the project.

2.1 Domain Name System

DNS (Domain Name System) is a critical system for translating human-readable domain names (like www.example.com) into IP addresses (like 93.184.216.34) that computers use to communicate over the internet. Often called the "phone book of the internet," DNS helps users reach websites and online resources by resolving domain names to their respective IP addresses.

2.1.1 How does DNS work When you type a URL in your browser, your device queries the DNS system to find the IP address associated with that domain. The request goes to a DNS resolver (usually provided by your ISP or a third party like Google or Cloudflare), which begins the process of finding the IP address by querying various DNS servers. The resolver first contacts a root name server. Root servers are the entry points to the DNS hierarchy and direct requests to the correct Top-Level Domain (TLD) servers based on the domain's suffix (e.g., .com, .org, .net). TLD servers handle requests for each top-level domain and direct the resolver to the authoritative name server for the domain, which stores the actual IP address and other DNS records. The authoritative server provides the IP address for the requested domain. This information is then sent back to the resolver and cached to speed up future requests for the same domain. The resolver returns the IP address to your device, which can now establish a direct connection to the server hosting the desired website.

2.1.2 Uses of DNS

- Resolving domain names for web browsing.

- Using MX records to direct email traffic.
- Content Delivery Networks use DNS to distribute user requests to the nearest server, improving load times.
- DNS can distribute requests across multiple servers to balance load and prevent overload on any single server.

2.2 DNS record types

There exist many types of the DNS records (which can be viewed here: https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4). However, the project's assignment demanded only support of the next seven DNS record types: - A - AAAA - SRV - SOA - MX - CNAME - NS

Therefore, only these types of records will be described below.

2.2.1 A Record (Address Record) Maps a domain name to an IPv4 address. If you query the **A** record for *example.com*, it might return *93.184.216.34*. Primarily used to point domain names to servers hosting websites or other internet services.

2.2.2 AAAA Record (IPv6 Address Record) Maps a domain name to an IPv6 address. If you query the **AAAA** record for *example.com*, it might return *2606:2800:220:1:248:1893:25c8:1946*. The usage is similar to an A record but for IPv6 addresses, supporting the newer IPv6 addressing scheme.

2.2.3 MX Record (Mail Exchange Record) Directs email to a mail server associated with the domain. An **MX** record for *example.com* might point to *mail.example.com* with a priority value. The priority value determines which server should be used first. Lower values indicate higher priority. These records are essential for setting up email services.

2.2.4 SOA Record (Start of Authority) Contains administrative information about a DNS zone. It has the primary name server, the email of the domain administrator, and timers related to zone refresh intervals. Every DNS zone must have an SOA record, which defines the primary authoritative server and settings for managing DNS updates and caching.

2.2.5 NS Record (Name Server Record) Specifies the authoritative name servers for a domain. An NS record for *example.com* could point to *ns1.example.com* and *ns2.example.com*. Indicates which servers hold the authoritative records for a domain, allowing DNS resolvers to reach the correct servers to answer queries for that domain.

2.2.6 SRV Record (Service Record) Specifies a service's location by defining the hostname and port number. Includes priority, weight, port, and target, allowing for load balancing and prioritization. Commonly used for services like VoIP, XMPP (Jabber), and other protocols that require a specific host and port to connect.

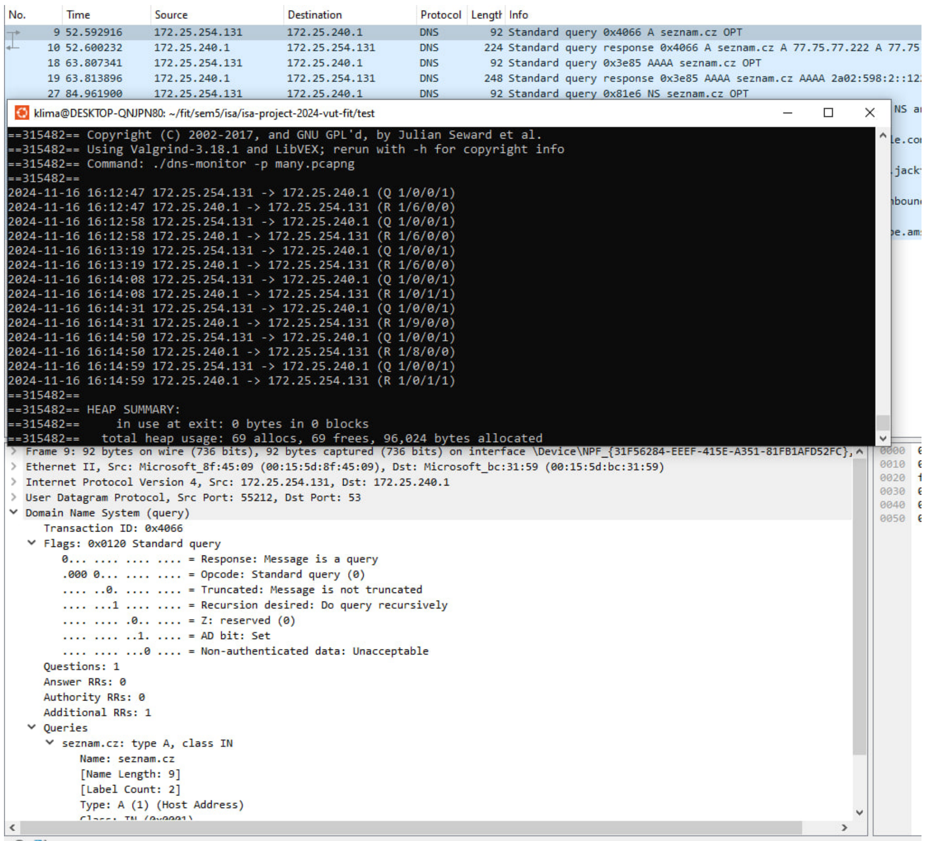
2.2.7 CNAME Record (Canonical Name Record) Maps one domain name to another, essentially creating an alias. If www.example.com has a CNAME pointing to example.com, it means that accessing www.example.com will retrieve records for example.com. Useful for pointing multiple domain names to a single IP without needing to update every domain's A record. CNAMEs allow a single host to have multiple names that all refer to the same resource.

3 Implementation design

According to the project's assignment, there are 2 kinds of program's output on **stdout**: simple and verbose. So I have created an abstract class *Packet_writer* and two classes *Simple_packet_writer* and *Verbose_packet_writer*. Until the DNS packet's question section, these two classes process a packet similarly, except they are providing different type of output. When a *Simple_packet_writer* reaches DNS packet's question section, next it only processes domain names and their translations if needed, ignoring all kinds of another records information. On the other hand, *Verbose_packet_writer* retrieves and outputs all information about each record.

4 Testing

I have created multiple **pcap** and **pcapng** files that contain DNS queries and responses of all types that my program is supposed to support and compared program's output to these files, and the result was the same. I opened these files using Wireshark program (https://www.wireshark.org/). The program properly handles memory access, allocation and deallocation so there must not be any memory leaks or any another kind of memory manipulation problems. I have used Valgrind program (https://valgrind.org/) for memory debugging and leak detection. I have also tested my program in the reference development environment (i.e. school servers **Eva** and **Merlin**). Below there are 2 examples of the comparison of the program's and Wireshark's output.

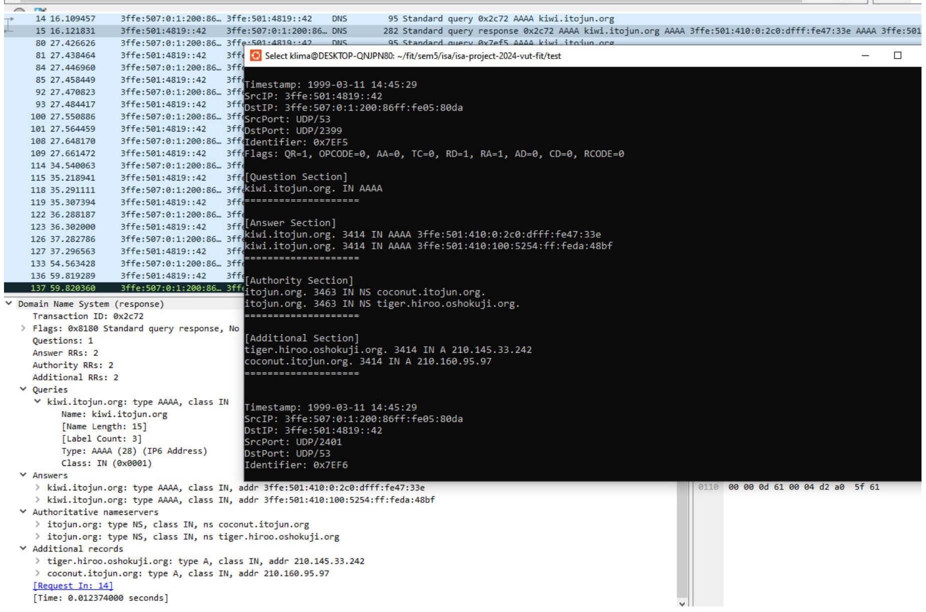


```
No.   Time         Source           Destination      Protocol Length Info
    9 52.592916    172.25.254.131   172.25.240.1     DNS      92 Standard query 0x4066 A seznam.cz OPT
   10 52.600232    172.25.240.1     172.25.254.131   DNS      224 Standard query response 0x4066 A seznam.cz A 77.75.77.222 A 77.75
   18 63.807341    172.25.254.131   172.25.240.1     DNS      92 Standard query 0x3e85 AAAA seznam.cz OPT
   19 63.813896    172.25.240.1     172.25.254.131   DNS      248 Standard query response 0x3e85 AAAA seznam.cz AAAA 2a02:598:2::12
   27 84.961900    172.25.254.131   172.25.240.1     DNS      92 Standard query 0x81e6 NS seznam.cz OPT
```

klima@DESKTOP-QNJPN80: ~/fit/sem5/isa/isa-project-2024-vut-fit/test

```
==315482== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==315482== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==315482== Command: ./dns-monitor -p many.pcapng
==315482==
2024-11-16 16:12:47 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:12:47 172.25.240.1 -> 172.25.254.131 (R 1/6/0/0)
2024-11-16 16:12:58 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:12:58 172.25.240.1 -> 172.25.254.131 (R 1/6/0/0)
2024-11-16 16:13:19 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:13:19 172.25.240.1 -> 172.25.254.131 (R 1/6/0/0)
2024-11-16 16:14:08 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:14:08 172.25.240.1 -> 172.25.254.131 (R 1/0/1/1)
2024-11-16 16:14:31 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:14:31 172.25.240.1 -> 172.25.254.131 (R 1/9/0/0)
2024-11-16 16:14:50 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:14:50 172.25.240.1 -> 172.25.254.131 (R 1/8/0/0)
2024-11-16 16:14:59 172.25.254.131 -> 172.25.240.1 (Q 1/0/0/1)
2024-11-16 16:14:59 172.25.240.1 -> 172.25.254.131 (R 1/0/1/1)
==315482==
==315482== HEAP SUMMARY:
==315482==     in use at exit: 0 bytes in 0 blocks
==315482==   total heap usage: 69 allocs, 69 frees, 96,024 bytes allocated
```

```
> Frame 9: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface \Device\NPF_{31F56284-EEEF-415E-A351-81FB1AFD52FC},
> Ethernet II, Src: Microsoft_8f:45:09 (00:15:5d:8f:45:09), Dst: Microsoft_bc:31:59 (00:15:5d:bc:31:59)
> Internet Protocol Version 4, Src: 172.25.254.131, Dst: 172.25.240.1
> User Datagram Protocol, Src Port: 55212, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x4066
  v Flags: 0x0120 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..1. .... = AD bit: Set
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  v Queries
    v seznam.cz: type A, class IN
        Name: seznam.cz
        [Name Length: 9]
        [Label Count: 2]
        Type: A (1) (Host Address)
```



```
 14 16.109457   3ffe:507:0:1:200:86… 3ffe:501:4819::42    DNS   95 Standard query 0x2c72 AAAA kiwi.itojun.org
 15 16.121831   3ffe:501:4819::42    3ffe:507:0:1:200:86… DNS   282 Standard query response 0x2c72 AAAA kiwi.itojun.org AAAA 3ffe:501:410:0:2c0:dfff:fe47:33e AAAA 3ffe:501
 80 27.426626   3ffe:507:0:1:200:86…
 81 27.438464   3ffe:501:4819::42
 84 27.446960   3ffe:507:0:1:200:86…
 85 27.458449   3ffe:501:4819::42
 92 27.470823   3ffe:507:0:1:200:86…
 93 27.484417   3ffe:501:4819::42
100 27.550886   3ffe:507:0:1:200:86…
101 27.564459   3ffe:501:4819::42
108 27.648170   3ffe:507:0:1:200:86…
109 27.661472   3ffe:501:4819::42
114 34.540063   3ffe:507:0:1:200:86…
115 35.218941   3ffe:501:4819::42
118 35.291111   3ffe:507:0:1:200:86…
119 35.307394   3ffe:501:4819::42
122 36.288187   3ffe:507:0:1:200:86…
123 36.302000   3ffe:501:4819::42
126 37.282786   3ffe:507:0:1:200:86…
127 37.296563   3ffe:501:4819::42
133 54.563428   3ffe:507:0:1:200:86…
136 59.819289   3ffe:501:4819::42
137 59.820360   3ffe:507:0:1:200:86…
```

Select klima@DESKTOP-QNJPN80: ~/fit/sem5/isa/isa-project-2024-vut-fit/test

```
Timestamp: 1999-03-11 14:45:29
SrcIP: 3ffe:501:4819::42
DstIP: 3ffe:507:0:1:200:86ff:fe05:80da
SrcPort: UDP/53
DstPort: UDP/2399
Identifier: 0x7EF5
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
kiwi.itojun.org. IN AAAA
====================

[Answer Section]
kiwi.itojun.org. 3414 IN AAAA 3ffe:501:410:0:2c0:dfff:fe47:33e
kiwi.itojun.org. 3414 IN AAAA 3ffe:501:410:100:5254:ff:feda:48bf
====================

[Authority Section]
itojun.org. 3463 IN NS coconut.itojun.org.
itojun.org. 3463 IN NS tiger.hiroo.oshokuji.org.
====================

[Additional Section]
tiger.hiroo.oshokuji.org. 3414 IN A 210.145.33.242
coconut.itojun.org. 3414 IN A 210.160.95.97
====================


Timestamp: 1999-03-11 14:45:29
SrcIP: 3ffe:507:0:1:200:86ff:fe05:80da
DstIP: 3ffe:501:4819::42
SrcPort: UDP/2401
DstPort: UDP/53
Identifier: 0x7EF6
```

```
v Domain Name System (response)
    Transaction ID: 0x2c72
  > Flags: 0x8180 Standard query response, No
    Questions: 1
    Answer RRs: 2
    Authority RRs: 2
    Additional RRs: 2
  v Queries
    v kiwi.itojun.org: type AAAA, class IN
        Name: kiwi.itojun.org
        [Name Length: 15]
        [Label Count: 3]
        Type: AAAA (28) (IP6 Address)
        Class: IN (0x0001)
  v Answers
    > kiwi.itojun.org: type AAAA, class IN, addr 3ffe:501:410:0:2c0:dfff:fe47:33e
    > kiwi.itojun.org: type AAAA, class IN, addr 3ffe:501:410:100:5254:ff:feda:48bf
  v Authoritative nameservers
    > itojun.org: type NS, class IN, ns coconut.itojun.org
    > itojun.org: type NS, class IN, ns tiger.hiroo.oshokuji.org
  v Additional records
    > tiger.hiroo.oshokuji.org: type A, class IN, addr 210.145.33.242
    > coconut.itojun.org: type A, class IN, addr 210.160.95.97
    [Request In: 14]
    [Time: 0.012374000 seconds]
```

```
0110  00 00 0d 61 00 04 d2 a0  5f 61
```

Bibliography

1. **RFC 2782 - A DNS RR for specifying the location of services (DNS SRV)**. Internet Engineering Task Force (IETF). Available at: https://datatracker.ietf.org/doc/html/rfc2782.
2. **DNS Parameters - Resource Record (RR) TYPEs**. Internet Assigned Numbers Authority (IANA). Available at: https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-2.
3. **RFC 1035 - Domain Names - Implementation and Specification**. Internet Engineering Task Force (IETF). Available at: https://datatracker.ietf.org/doc/html/rfc1035.
4. **RFC 3596 - DNS Extensions to Support IP Version 6**. Internet Engineering Task Force (IETF). Available at: https://datatracker.ietf.org/doc/html/rfc3596#page-2.
5. **RFC 2065 - Domain Name System Security Extensions**. Internet Engineering Task Force (IETF). Available at: https://datatracker.ietf.org/doc/html/rfc2065#page-30.
6. **Matoušek, P.** *Síťové aplikace a jejich architektura*. VUTIUM, 2014.
7. **Project 1 Primer - DNS and HTTP**. Mislove.org. Available at: https://mislove.org/teaching/cs4700/spring11/handouts/project1-primer.pdf.