VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

SÍŤOVÉ APLIKACE A SPRÁVA SÍTÍ

2019/2020

Projekt

DNS resolver

Jakub Sadílek (xsadil07) Brno, 18. listopadu 2019

Obsah

Úvod

Zadání projektu bylo vytvořit program **dns**, který bude umět zasílat dotazy na zadaný DNS server a v čitelné podobě vypisovat přijaté odpovědi. Sestavení a analýza paketů musí být implementována přímo v programu dns. Program by měl být co nejvíce multiplatformní na operačních systémech založených na unixu.

Implementace

Program je implementován v programovacím jazyce C++ ve standardu C++14, jelikož například používám zápis čísel v binární podobě a tenhle zápis i když funguje v předešlých verzích, standardizován byl až v **std14**. Komunikace je realizována přes UDP protokol. Pokud uživatel přímo nezadá cílový port, defaultně je nastavený port 53, protože je určený pro DNS komunikaci. Dále program podporuje verze IP protokolu IPv4 a IPv6.

Zpracování argumentů

Jako první krok program zkontroluje a zpracuje vstupní parametry. To je realizováno ve funkci `parseArgs()`. Jelikož program podporuje poměrně velké množství parametrů bylo by výhodné použít dostupné funkce pro jejich zpracování, ovšem já se rozhodl jít cestou podmínek a cyklů.

V cyklu jsou procházeny postupně všechny argumenty a pokud program narazí na známý přepínač, tak se rozhodneme o další akci, tj. poznačení rekurze, reverzního dotazu či AAAA dotazu pro pozdější zpracování, uložení cílového portu, zpracování adresy dotazovaného serveru nebo zpracování dotazované adresy, pokud nepředcházel žádný přepínač. Všechny tyto informace jsou uloženy do globální struktury, aby k nim mohli později bez problému přistoupit i ostatní části programu.

Kontrola vstupního portu je realizována ve funkci `validatePort()`, která zkontroluje jestli se v hodnotě nenachází nepovolené znaky (například písmena) a validuje povolený rozsah tj. 1-65535. Zpracování adresy dotazovaného serveru je implementována ve funkci `getHostIP()`, kde je veškerá povinnost přenechána na funkci `getaddrinfo()` z knihovny netdb.h, která nejen, že validuje zápis adresy, ale v případě doménového jména nám ho převede na IP adresu, kterou si uložíme. Zpracování dotazované adresy může proběhnout třemi způsoby. V případě, že se jedná o IPv4 adresu, tak adresu otočíme, ale rámce 1 bytu zůstanou zachovány a nakonec připojíme řetězec „.in-addr.arpa“ – jde o tzv. reverzní dotaz na adresu, podle které DNS vyhledá odpovídající doménové jméno. Podobně je to s IPv6 adresou, tu otočíme taky po bytech, ale k tomu ještě otočíme horní a spodní 4 bity v každém bytu a tyto 4 bity oddělíme tečkou a nakonec přidáme řetězec „.in6.arpa“. Jednoduše musíme adresu otočit tak, aby hexadecimální zápis byl opačný (přitom jedno hexadecimální číslo má velikost 4 bity). Poslední způsob je zpracování je doménového jména, který není nijak zajímavý, protože ho nijak nekontroluji a nechám odpověď na DNS serveru, tj. v případě neexistujícího jména nám nedojde maximálně žádná odpověď. Jako poslední věc funkce zkontroluje neplatnou kombinaci parametru jako je „-x“ a „-6“, protože

nedává smysl si vyžadovat dotaz typu AAAA, když požadujeme reverzní dotaz nebo „-x“ se zadanou IP adresou, protože reverzní dotaz požadujeme na doménu.

Sestavení dotazu

Poté co máme zpracované parametry sestavíme náš požadavek a odešleme ho DNS serveru. Celé odeslání a přijmutí odpovědi je realizováno ve funkci `sendAndReceive()`. Celé sestavení a analýzu budeme provádět v bufferu jehož délka je nastavena na 65536 bytu, protože je to dost bezpečná délka na přijetí odpovědi ze serveru i na náš dotaz (všech 65536 bytu ovšem posílat nebudeme).

Na začátku vytvoříme strukturu IP adresy. Sockaddr_in pro IPv4 nebo sockaddr_in6 pro IPv6, podle adresy zadaného serveru, kam pošleme náš dotaz a naplníme ji adresou a portem, které už máme zpracované z předchozí části. Inicializujeme socket pro odeslání **UDP** a naplníme buffer. Na začátek vložíme **DNS hlavičku** a naplníme ji. O to se postará funkce `fillDNShdr()`, která ID nastaví podle proces ID (zkr. PID), RD flag nastaví podle argumentu rekurze a počet otázek (QCOUNT) nastaví na 1, protože odešleme pouze jeden dotaz. Zbytek parametrů v DNS hlavičce nastavíme na 0, protože pro nás **aktuálně** nejsou zajímavé nebo je nepotřebujeme. Dále do bufferu vložíme data týkající se naší otázky, tedy **adresu serveru** (QNAME), o to se postará funkce `formatQNAME()`, která převede jméno/adresu na **DNS formát** tj. před každý kousek řetězce napíšeme číslo odpovídající počtu znaků v tomto podřetězci, přitom jednotlivé kusy řetězce jsou odděleny tečkou - například www.google.com převede na 3www6google3com (pozn. všimněme si, že jméno může mít libovolnou délku). Následuje vložení dalších dat týkající se naší otázky, a to je **typ** a **třída** (které už mají pevně danou velikost). Zavoláme funkci `fillQuestionData()`, která data nastaví, přesněji typ nastaví na A v případě dotazu na IPv4 adresu, AAAA v případě dotazu na IPv6 adresu nebo PTR v případě reverzního dotazu. Třídu nastavíme na IN, jelikož se jedná o internet.

Poté co máme sestavený dotaz můžeme data odeslat přes funkci `sendto()` a odešleme pouze kus bufferu, který jsme vyplnili, abychom nezatěžovali síť (typicky pod 100 bytu). Data dorazí na server a v lepším případě nám odpoví, v horším případě nám neodpoví, to se může stát, když pošleme dotaz na nějaké zařízení, které není určeno proto, aby nám odpovědělo, nebo na jiný port, na kterém zrovna DNS nenaslouchá, proto musíme naše čekání ošetřit **timeoutem**, který je nastaven na 3 sekundy a je realizován přes systémový signál SIGALRM, který po vypršení časového intervalu zavolá funkci `alarm_handler()`, která uzavře socket, vypíše chybovou hlášku a ukončí program. Pokud odpověď přijmeme, uložíme si ji do našeho bufferu (přepíšeme jim náš předchozí dotaz, který už nebudeme potřebovat).

Analýza odpovědi

Nyní jsme v situaci, kdy jsme odeslali náš požadavek na server, který nám odpověděl a my si odpověď uložili do bufferu. O celou analýzu se postará funkce `decodeMessage()`, ve které si inicializuje pointer jako tzv. **reader**, který bude plnit funkci čtecí hlavy, se kterou

se budeme pohybovat po paketu a číst z něj informace, které si budeme postupně ukládat do globální proměnné, kterou na konci vypíšeme jako výsledek.

Jako první přečteme **DNS hlavičku**, která se nachází na začátku bufferu. Odtud nás zajímá informace autoritativní, rekurzivní a zkrácený flag, jak je specifikováno v zadání. Tedy je přečteme a podle jejich pravdivostní hodnoty uložíme jejich stav a posuneme pointer za DNS hlavičku. Nyní dekódujeme další sekce, které se v paketu nacházejí, tj. sekce s otázkou (kterou jsme si v předchozí sekci sestavili tzv. **question section**), odpovědí (**answer section**), autoritou (**authority section**) a další sekcí (**additional section**). O zpracování sekce s otázkou se postará funkce `decodeQuestion()`, která zpracuje jméno, typ a třídu (viz dále). Před každou sekcí ještě vypíšeme informaci, kolik záznamů se v dané sekci nachází. Tato informace je uložena v DNS hlavičce (posledních 64 bytu přitom každá informace má velikost 16 bytu). Výhoda je, že tyto další tři sekce mají stejný formát, takže na jejich analýzu nám stačí implementovat pouze jednu funkci a stačí ji pouze zavolat s jinými daty. To je realizováno ve funkci `decodeQuery()`, která přijímá první parametr náš buffer, počet záznamů v dané sekci a ukazatel na začátek dané sekce (náš čtecí pointer). Funkce se dokonce postará sama i o posunutí pointeru, takže po analýze bude pointer ukazovat přesně na konec dané sekce, a to se nám velmi hodí, protože ji můžeme třikrát pohodlně zavolat a bude provedena celá analýza.

Pojďme se se detailně podívat na implementaci analýzy dané sekce. Funkce `decodeQuery()` je jeden velký cyklus s předem známým počtem opakování, který cyklí s každým záznamem a ten dekóduje. Jako první se vždy nachází **jméno** s proměnnou velikostí, které je potřeba dešifrovat. To je implementováno ve funkci `decodeName()`, která jméno vrátí jako string a my s ním můžeme dále pracovat (resp. uložit) a automaticky posune náš pointer za jméno. Stručně si popišme postup, jak funkce pracuje. Jelikož doménové jméno může být dlouhé a zbytečně by zvětšovalo velikost paketu a tím zahlcovalo síť, je možné, že pokud se už v paketu část tohoto jména vyskytuje, jednoduše se na něj můžeme odkázat a pokračovat se čtením na onom daném místě. Takže budeme postupně číst každý normální znak a pokud bude potřeba skočit, tak se přesuneme a budeme pokračovat se čtením na daném místě. Ale jak poznat kdy skočit nebo kam? To je dáno binární hodnotou znaku (pro lepší přehlednost jsem v kódu použil binární zápis, proto std14). Pokud znak má binární hodnotu, kde první 2 nejvyšší bity jsou nastaveny na „11“ je to pro nás znamení, že se musíme přesunout a offset kam, je daný 2 byty přesněji aktuální znakem (8 bytu) a následujícím (dalších 8 bytu), ale první 2 jsou náš indikátor, takže offset je posledních 14 bitů. Posun je pak dán od začátku paketu, takže stačí vzít paket a přičíst offset. Poté co přečteme všechny znaky (narazíme na nultý znak pro ukončení řetězce) stačí jméno dekódovat z DNS formátu na klasický čitelný, tj. 3www6google3com na www.google.com. A provést korekci čtecího pointeru, aby ukazoval na konec místa se jménem, a nikoliv postával někde v paketu. (pozn. pokud doménové jméno je dáno pouze nultým znakem (má nulovou délku) jedná se o <*Root*>). Poté co máme zpracované jméno, se v bufferu nachází záznam o **typu** a **třídě** záznamu. Rozpoznávané záznamy jsou: A, NS, CNAME, SOA, WKS, PTR, MX, TXT, AAAA, SRV, ANY. Přestože v zadání je definováno, že program se musí vypořádat se záznamy typu CNAME, rozhodl jsem se plně podporovat záznamy typy A, AAAA, CNAME, PTR, NS a SOA (můžeme chápat jako **rozšíření** nad rámec zadání). Dále v bufferu se nachází záznam o třídě, kde rozpoznáme IN, CH, HS, ANY. Dále je poznačeno **TTL** a **délka dat**, pak následují samotná **data**.

Nyní si stručně popíšeme analýzu jednotlivých typů, které podporujeme (viz. výše). U záznamu typu „A“ očekáváme v datech uloženou IPv4 adresu, takže není nic jednoduššího, než na číslo pohlídnout jako na 4 byty (protože IPv4 adresa má délku 32 bitů) překonvertovat na adresu pomocí funkce `inet_ntoa()`, uložit jako string a posunout pointer za tyto data (viz. předchozí záznam). Záznam typu AAAA je víceméně to samé s IPv6 adresou, akorát ta má délku 128 bitů a pro převod z binární podoby použijeme funkci `inet_ntop()`. Další záznamy jsou CNAME, PTR a NS. Tyto typy mají stejnou strukturu, takže analýza bude probíhat stejně. V datech je uloženo **doménové jméno**, takže zavoláme naší funkci `decodeName()` (viz. popis výše), která jméno převede a my ho jen uložíme. Posledním podporovaným záznamem je typ SOA, což je poměrně velká struktura. Obsahuje jméno primárního serveru, adresu elektronické pošty jejího správce (zavináč je nahrazen tečkou), sériové číslo, refresh, retry, expire a TTL. První dva záznamy, tj. jméno serveru a mail jsou opět jména s proměnnou délkou pro naší funkci `decodeName()` a zbytek normálně přečteme (namapujeme strukturu a přistoupíme k jednotlivým položkám).

Výpis

Velká část programu potřebuje přímo zapisovat důležité informace z analýzy na výstup od flagů v DNS hlavičce, přes počet jednotlivých záznamů až po jednotlivé záznamy. To je realizováno globálním stringem, do kterého postupně tyto informace ukládáme a na konci ho vypíšeme. Další výhodu má, že v případě chyby nejsou na standardním výstupu napůl vypsaná data. Pokud nastane chyba je na **standardní chybový výstup** vypsána chybová hláška a program je ukončen návratovou hodnotou **1**.

Testování

1) Rekurzivní dotaz na doménu

```
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s 8.8.8.8 www.ietf.org -r
Authoritative: No, Recursive: Yes, Truncated: No
Question section (1)
www.ietf.org, A, IN
Answer section (3)
www.ietf.org, CNAME, IN, 9, www.ietf.org.cdn.cloudflare.net
www.ietf.org.cdn.cloudflare.net, A, IN, 265, 104.20.0.85
www.ietf.org.cdn.cloudflare.net, A, IN, 265, 104.20.1.85
Authority section (0)
Additional section (0)
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	DNS	72	Standard query 0x0ed7 A www.ietf.o
2	0.020321547	8.8.8.8	10.0.2.15	DNS	149	Standard query response 0x0ed7 A w

```
▾ Domain Name System (response)
    Transaction ID: 0x0ed7
  ▾ Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ www.ietf.org: type A, class IN
  ▾ Answers
    ▾ www.ietf.org: type CNAME, class IN, cname www.ietf.org.cdn.cloudflare.net
        Name: www.ietf.org
        Type: CNAME (Canonical NAME for an alias) (5)
        Class: IN (0x0001)
        Time to live: 9
        Data length: 33
        CNAME: www.ietf.org.cdn.cloudflare.net
    ▾ www.ietf.org.cdn.cloudflare.net: type A, class IN, addr 104.20.0.85
        Name: www.ietf.org.cdn.cloudflare.net
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 265
        Data length: 4
        Address: 104.20.0.85
    ▾ www.ietf.org.cdn.cloudflare.net: type A, class IN, addr 104.20.1.85
        Name: www.ietf.org.cdn.cloudflare.net
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 265
        Data length: 4
        Address: 104.20.1.85
    [Request In: 1]
    [Time: 0.020321547 seconds]
```

2) Reverzní dotaz na IPv6 adresu

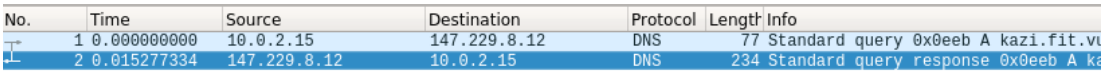
```
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s 8.8.8.8 2620:119:35::35 -x
Authoritative: No, Recursive: No, Truncated: No
Question section (1)
5.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.3.0.0.9.1.1.0.0.2.6.2.ip6.arpa, PTR, IN
Answer section (1)
5.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.3.0.0.9.1.1.0.0.2.6.2.ip6.arpa, PTR, IN, 233, resolver1.opendns.com
Authority section (0)
Additional section (0)
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	DNS	132	Standard query 0x0ece PTR 5.3.0.0.0.0.0.0…
2	0.019815597	8.8.8.8	10.0.2.15	DNS	167	Standard query response 0x0ece PTR 5.3.0.0.…

```
▸ Frame 2: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_68:97:c6 (08:00:27:68:97:c6)
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 10.0.2.15
▸ User Datagram Protocol, Src Port: 53, Dst Port: 40444
▾ Domain Name System (response)
    Transaction ID: 0x0ece
  ▾ Flags: 0x8080 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...0 .... .... = Recursion desired: Don't do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ 5.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.3.0.0.9.1.1.0.0.2.6.2.ip6.arpa: type PTR, class IN
  ▾ Answers
    ▾ 5.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.3.0.0.9.1.1.0.0.2.6.2.ip6.arpa: type PTR, class IN, resolver1.opendns.com
        Name: 5.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.3.0.0.9.1.1.0.0.2.6.2.ip6.arpa
        Type: PTR (domain name PoinTeR) (12)
        Class: IN (0x0001)
        Time to live: 233
        Data length: 23
        Domain Name: resolver1.opendns.com
    [Request In: 1]
    [Time: 0.019815597 seconds]
```

3) Autoritativní dotaz

```
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s kazi.fit.vutbr.cz kazi.fit.vutbr.cz
Authoritative: Yes, Recursive: No, Truncated: No
Question section (1)
kazi.fit.vutbr.cz, A, IN
Answer section (1)
kazi.fit.vutbr.cz, A, IN, 14400, 147.229.8.12
Authority section (4)
fit.vutbr.cz, NS, IN, 14400, rhino.cis.vutbr.cz
fit.vutbr.cz, NS, IN, 14400, kazi.fit.vutbr.cz
fit.vutbr.cz, NS, IN, 14400, gate.feec.vutbr.cz
fit.vutbr.cz, NS, IN, 14400, guta.fit.vutbr.cz
Additional section (2)
guta.fit.vutbr.cz, A, IN, 14400, 147.229.9.11
guta.fit.vutbr.cz, AAAA, IN, 14400, 2001:67c:1220:809::93e5:90b
```

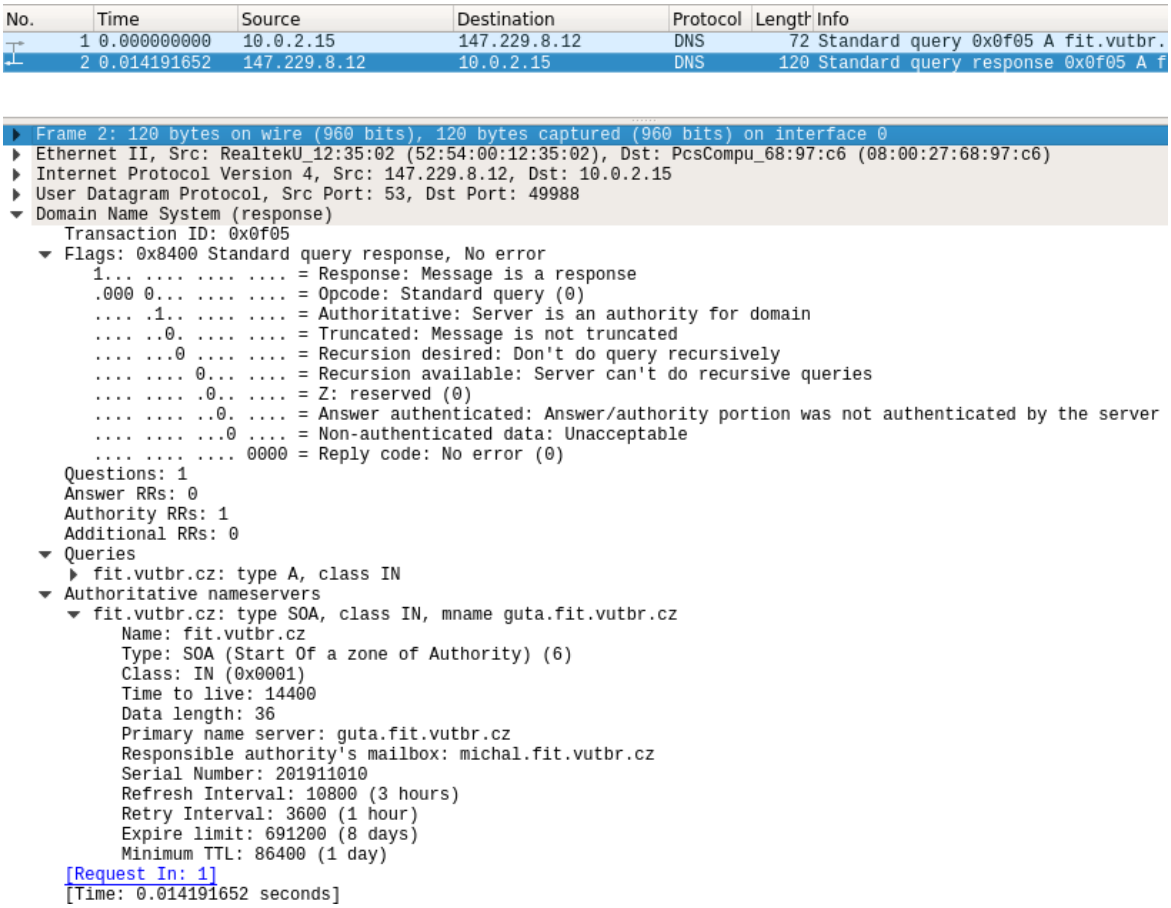


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	147.229.8.12	DNS	77	Standard query 0x0eeb A kazi.fit.vu
2	0.015277334	147.229.8.12	10.0.2.15	DNS	234	Standard query response 0x0eeb A ka

```
▸ Frame 2: 234 bytes on wire (1872 bits), 234 bytes captured (1872 bits) on interface 0
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_68:97:c6 (08:00:27:68:97:c6)
▸ Internet Protocol Version 4, Src: 147.229.8.12, Dst: 10.0.2.15
▸ User Datagram Protocol, Src Port: 53, Dst Port: 36844
▾ Domain Name System (response)
    Transaction ID: 0x0eeb
  ▾ Flags: 0x8400 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .1.. .... .... = Authoritative: Server is an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...0 .... .... = Recursion desired: Don't do query recursively
      .... .... 0... .... = Recursion available: Server can't do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 4
    Additional RRs: 2
  ▾ Queries
    ▸ kazi.fit.vutbr.cz: type A, class IN
  ▾ Answers
    ▸ kazi.fit.vutbr.cz: type A, class IN, addr 147.229.8.12
  ▾ Authoritative nameservers
    ▸ fit.vutbr.cz: type NS, class IN, ns rhino.cis.vutbr.cz
    ▸ fit.vutbr.cz: type NS, class IN, ns kazi.fit.vutbr.cz
    ▸ fit.vutbr.cz: type NS, class IN, ns gate.feec.vutbr.cz
    ▸ fit.vutbr.cz: type NS, class IN, ns guta.fit.vutbr.cz
  ▾ Additional records
    ▸ guta.fit.vutbr.cz: type A, class IN, addr 147.229.9.11
    ▸ guta.fit.vutbr.cz: type AAAA, class IN, addr 2001:67c:1220:809::93e5:90b
    [Request In: 1]
    [Time: 0.015277334 seconds]
```

4) SOA záznam

```
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s kazi.fit.vutbr.cz fit.vutbr.cz
Authoritative: Yes, Recursive: No, Truncated: No
Question section (1)
fit.vutbr.cz, A, IN
Answer section (0)
Authority section (1)
fit.vutbr.cz, SOA, IN, 14400, guta.fit.vutbr.cz
        Responsible Authority's mailbox: michal.fit.vutbr.cz
        Serial number: 201911010
        Refresh interval: 10800 seconds
        Retry interval: 3600 seconds
        Expire limit: 691200 seconds
        Minimum TTL: 86400 seconds
Additional section (0)
```



5) Timeout a chybové výstupy

```
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s kazi.fit.vutbr.cz -p 10 fit.vutbr.cz
Nepodarilo se zachytit zadnou odpoved.
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ echo $?
1
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -x -s kazi.fit.vutbr.cz fit.vutbr.cz
Pro reverzni dotaz je treba zadat validni IP adresu.
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ echo $?
1
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s kazi.fit.vutbr.cz 8.8.8.8 -x -6
Neplatna kombinace parametru -x a -6.
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ echo $?
1
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ ./dns -s kazi.fit.vutbr.cz 8.8.8.8
Pro dotaz na domenu je potreba pouzit reverzni dotaz (-x).
izuwei@izuwei-VirtualBox:~/Plocha/ISA$ echo $?
1
```

Závěr

Líbí se mi praktická použitelnost programu a možná proto mě i jeho implementace bavila. Také jsem rád, že se mi povedlo implementovat IP protokol IPv6. Obtížnost byla podle mě přiměřená a moje zkušenosti určitě projekt obohatil.

Použitá literatura a zdroje

- RFC 1035 - https://tools.ietf.org/html/rfc1035
- RFC 3596 - https://tools.ietf.org/html/rfc3596
- SIGALRM - https://stackoverflow.com/questions/4583386/listening-using-pcap-with-timeout
- Struktury - https://www2.cs.duke.edu/courses/fall16/compsci356/DNS/DNS-primer.pdf