VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

DNS RESOLVER

AUTOR PRÁCE **ADRIÁN PONECHAL**
AUTHOR

BRNO 2023

Obsah

Kapitola 1

Úvod

Komunikácia je neodmysliteľnou súčasťou zdielania a prenosu informácii. V súčasnosti sa na tento účel využíva hlavne výpočtová technológia.

Základ počítačovej komunikácie tvoria dve hlavné zložky: adresovanie a smerovanie. Adresovanie je spôsob vytvárania a priraďovania adries počítačom. Adresa je jednoznačný údaj, ktorý presne identifikuje práve jeden adresovateľný prvok. Smerovanie je proces výberu cesty pre prevádzku v sieti medzi viacerými sieťami. [2]

Adresa zariadení na úrovni TCP/IP je vo formáte IP adresy (napríklad 95.82.140.207). Pre človeka je adresovanie na úrovni počítačov komplikované. Adresovanie počítačov pomocou IP adries nie je nutne dobré riešenie pre užívateľov, pretože sú náročné na zapamätanie. Z toho dôvodu bol vyvinutý systém, ktorý umožňuje adresovať zariadenia pomocou doménových adries.

1.1 Domain name system

DNS (angl. domain name system) je systém, ktorého cieľom je poskytnúť mechanizmus na pomenovávanie zdrojov (zariadení) takým spôsobom, aby tieto mená boli použiteľné v rôznych zariadeniach, sieťach, protokolových rodinách, na internete a administratívnych organizáciach. [5]

1.2 Architektúra DNS

Základnou úlohou služby DNS je mapovanie doménových adries (tzv. doménových mien) na IP adresy. Systém DNS sa skladá z troch hlavných častí - priestoru doménových mien, DNS serverov a resolveru. Priestor doménových mien je databáza, kde sú dané doménové mená uložené. Táto databáza je hierarchicky usporiadaná do stromovej štruktúry (pre rýchle vyhladávanie konkrétnych domén). [4]

1.3 DNS resolver

Je to klientský program, ktorý získavá informácie zo systému DNS prostredníctvom dotazovania sa na DNS servery. Tento proces sa nazýva DNS rezolúcia. Resolver môže mať viac typov konfigurácii. Táto práca je zameraná na nárvh a implementáciou tzv. stub resolvera.

Kapitola 2

Návrh programu

2.1 Stub resolver

Stub resolver je typ resolvera, ktorý interaguje s aplikáciou alebo užívateľom a rekurzívnym DNS serverom. Samotný resolver rezolúciu nevykonáva. V tomto prípade sa rezolúcia realizuje zaslaním dotazu resolvera na rekurzívny DNS server, ktorý odošle odpoveď na dotaz. Prijatú odpoveď resolver spracuje a získané informácie poskytne aplikácii alebo zobrazí užívateľovi.

2.2 Dotazovanie

Dotazovanie na server prebieha zasielaním správ. Užívateľ alebo aplikácia poskytne údaje resolveru (ako sú dotazovaná adresa, typ dotazu). Resolver dané údaje vloží do správy, ktorú odošle na DNS server, na čo DNS server odpovie. Komunikácia prebieha štandartne cez protokol UDP. Jeden dotaz zodpovedá jednému UDP datagramu.

2.2.1 Formát správy

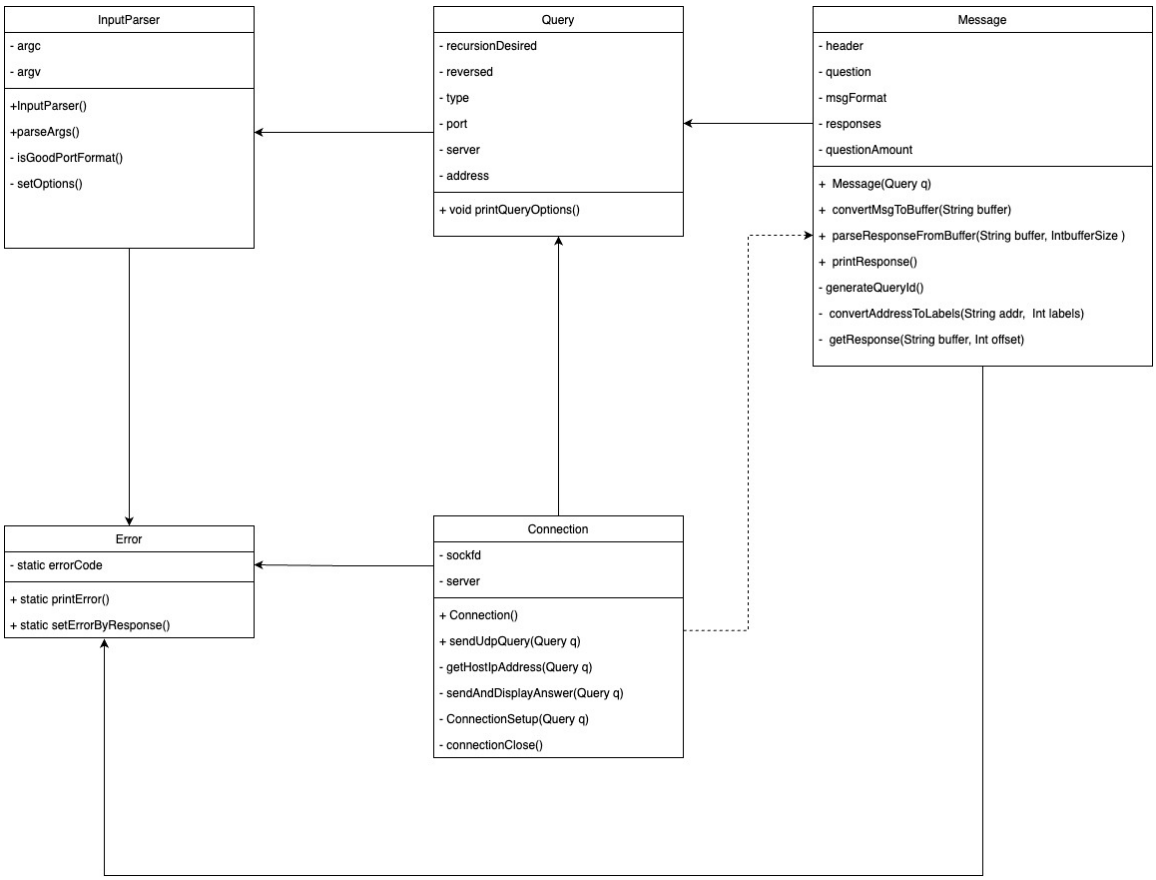
Správa sa skladá z 5 hlavných častí: hlavičky, otázky, odpovednej sekcie, autoritatívnej sekcie a dodatočnej sekcie. V hlavičke sú uložené údaje o dotaze, ako ID, typ dotazu, príznaky (dodatočné informácie o správe), kód chyby a čítače záznamov pre jednotlivé sekcie. Časť otázky sa skladá z dotazovanej adresy rozloženej na časti (tzv. labels), typu dotazu a triedy dotazu (typicky IN pre internet).

Príznaky

- QR - tento príznak odlišuje dotaz (0) a odpoveď (1)
- OPCODE - typ dotazu
- AA - ak je nastavený na hodnotu 1, tak odpoveď je autoritatívna (dotazovaný server je autorita pre dotazovanú doménu)
- TC - ak je nastavený na hodnotu 1, tak odpoveď je skrátená
- RD - ak je nastavený na hodnotu 1, tak je požadovaná rekurzívna rezolúcia
- RA - určuje, či je rekurzívna rezolúcia implementovaná v odpovedi
- Z - príznak rezervovaný pre budúce použitie (musí byť nulový)
- RCODE - kód odpovede

Príznaky predstavujú základné informácie o obsahu správy. Pri zasielaní dotazu sa nastavujú parametre v hlavičke a otázkovej sekcii. Zvyšné sekcie sú určené pre zaznamenanie odpovede. Hodnoty príznakov QR, AA, TC, RA, Z, RCODE musia byť pri zaslaní dotazu nulové.

2.3 Objektový nárvh



Obr. 2.1: Class diagram

Program sa skladá z 5 hlavných tried: InputParser, Query, Message, Connection a Error. Prvé 4 triedy spravujú správy a trieda Error spracováva chybové stavy, chybové hlásenia a návratovú hodnotu programu.

Trieda InputParser má za úlohu spracovať argumenty zadané používateľom a vytvoriť dotaz. Dotaz reprezentuje trieda Query. Spracovanie dotazu, jeho zaslanie a prijatie odpovede má za úlohu trieda Connection. Trieda Message slúži na vytvorenie správy z dotazu a jej konverziu medzi formátom, ktorý sa odosiela po sieti a formátom, ktorý spracováva aplikácia.

Kapitola 3

Popis programu

3.1 Vstup

Program príjma vstupy z príkazovej riadky pri spustení.

```
./dns [-r] [-x] [-6] -s server [-p port] address
```

Popis prepínačov:

- -r nastaví príznak pre rekurzívny dotaz
- -x nastaví príznak pre reverzný dotaz
- -6 nastaví príznak pre dotazovanie sa na adresy IPv6
- -s povinný prepínač; za ním nasleduje adresa serveru
- server je adresa serveru, na ktorý budeme posielať dotaz
- -p nepovinný prepínač; nastaví číslo portu
- address je adresa, na ktorú sa budeme dotazovať

V prípade použitia nedefinovaného prepínača program vypíše chybové hlásenie a program skončí. V prípade viacerých zadaných prepínačov -s program použije poslednú zadanú adresu serveru. To isté platí pre zadávanie adries. Ak je pri zadávaní portu zadaný port v neplatnom formáte, program použije štandartný port pre DNS (port číslo 53).

3.2 Výstup

Výstup z programu je vo formáte

```
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)
```

Kde AUTHORITATIVE: NO značí, že dotazovaný server nie je autorita pre danú doménu, RECURSIVE: YES značí, či bol odpoveď nájdená rekurzívne, a TRUNCATED: NO značí, že odpoveď nebola skrátená. V prípade skrátenej odpovede vypíše program na výstup túto informáciu sekciu otázky. Zvyšné sekcie program nevypíše.
Štandartný formát odpovede: <dotazovaná doména>, <typ záznamu>, <trieda dotazu>, <ttl>, <dáta odpovede>.

Návratové hodnoty:

- 0 : úspech
- 1 - 5 : kód chyby získanej od serveru
- 6 : zlyhanie spojenia
- 7 : zlé vstupné parametre
- 99 : interná chyba programu

3.3 Príklad spustenia

Príklad rekurzívneho spustenia:

```
./dns -r -s dns.google.com www.fit.vut.cz
```

Výstup:

```
Authoritative: No, Recursive: Yes, Truncated: No
Question section (1)
  www.fit.vut.cz., A, IN
Answer section (1)
  www.fit.vut.cz., A, IN, 12283, 147.229.9.26
Authority section (0)
Additional section (0)
```

Príklad nerekurzívneho spustenia

```
./dns -s kazi.fit.vutbr.cz www.vut.fit.cz
```

Výstup:

```
Authoritative: No, Recursive: No, Truncated: No
    Question section (1)
      www.vut.fit.cz., A, IN
    Answer section (0)
    Authority section (4)
      cz., NS, IN, 172800, b.ns.nic.cz.
      cz., NS, IN, 172800, c.ns.nic.cz.
      cz., NS, IN, 172800, d.ns.nic.cz.
      cz., NS, IN, 172800, a.ns.nic.cz.
    Additional section (8)
      d.ns.nic.cz., A, IN, 172800, 193.29.206.1
```

```
c.ns.nic.cz., A, IN, 172800, 194.0.14.1
b.ns.nic.cz., A, IN, 172800, 194.0.13.1
a.ns.nic.cz., A, IN, 172800, 194.0.12.1
d.ns.nic.cz., AAAA, IN, 172800, 2001:678:1:0:0:0:0:1
c.ns.nic.cz., AAAA, IN, 172800, 2001:678:11:0:0:0:0:1
b.ns.nic.cz., AAAA, IN, 172800, 2001:678:10:0:0:0:0:1
a.ns.nic.cz., AAAA, IN, 172800, 2001:678:f:0:0:0:0:1
```

3.4 Zaujímavé časti kódu

3.4.1 Vytváranie spojenia

Pre vytovernie spojenia je treba získať najprv adresu zo serveru. Získavanie adresy serveru je realizovná prostredníctvom vstavanej funkcie **gethostbyname()**.

Použitie funckie gethostbyname():

```
// check if host was found
hostInfo = gethostbyname(server);
if (hostInfo == NULL)
{
    Error::printError(CONNECTION_FAILED, "Host IP address not found.\n");
    return false;
}

// get ip address
struct in_addr **addressList = (struct in_addr **)hostInfo->h_addr_list;
if (addressList == NULL)
{
    Error::printError(CONNECTION_FAILED, "Host IP address not found.\n");
    return false;
}
```

Získanie adresy z vráteného listu adries:

```
for (size_t i = 0; addressList[i] != NULL; i++)
{
    char *ipAddr = inet_ntoa(*addressList[i]);
    if (inet_pton(AF_INET, ipAddr, &dst->s_addr) < 1)
    {
        // if there is no more ip address in list, get error
        if (addressList[i + 1] == NULL)
        {
            Error::printError(CONNECTION_FAILED, "Host IP address
            not found.\n");
            return false;
        }
    }
    // found
    else
    {
        break;
    }
}
```

Obe časti kódu su prevzaté zo stránky GeeksForGeeks [6].

Po získaní adresy môžeme prejsť k zahájeniu sponia. Samotné spojenie je realizované pomocou UDP schránok:

Príprava spojenia

```
this->sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0)
{
    Error::printError(CONNECTION_FAILED, "socket() failed\n");
    return false;
}

// setup server address
memset(&this->server, 0, sizeof(sockaddr_in));

struct in_addr serverAddr;
if (!getHostIPaddress(query.getServer().c_str(), &serverAddr))
{
    return false;
}

this->server.sin_addr = serverAddr;
this->server.sin_port = htons(query.getPort());
this->server.sin_family = AF_INET;
```

Odoslanie query

```
if (connect(this->sockfd, (struct sockaddr *)(&this->server),
sizeof(this->server)) < 0)
{
    Error::printError(CONNECTION_FAILED, "connect() failed\n");
    return;
}

...

int bytesTx = send(this->sockfd, (const char *)buffer,
bufferLength, 0);
if (bytesTx < 0)
{
    Error::printError(CONNECTION_FAILED, "send() failed\n");
    return;
}
```

Príjmanie odpovede

```
char recvBuffer[UDP_DATAGRAM_LIMIT] = {0};
int bytesRx = recv(this->sockfd, (char *)recvBuffer,
UDP_DATAGRAM_LIMIT, 0);
if (bytesRx < 0)
{
    Error::printError(CONNECTION_FAILED, "recv() failed\n");
    return;
}
```

Zdroj: [3]

Timeout

Pre prípad, že server nepošle odppoveď, alebo služba na danom serveri nebeží, je v tomto resolveri implementovaný timeout.[1]

```
struct timeval timeout;
timeout.tv_sec = 2;
timeout.tv_usec = 0;
setsockopt(this->sockfd, SOL_SOCKET, SO_RCVTIMEO,
(const char *)&timeout, sizeof(timeout));
```

3.5 Kompatibilita

Program bol vyvíjaný pomocou jazyka C++. Pre preklad bol použitý nástroj GNU Make 3.81. Preklad bol prekladaný pomocou štandardu c++17. Program Kompatibilný s operačnými systémami typu UNIX.

Kapitola 4

Testovanie

4.1 Spôsoby testovania

4.2 Použité nástroje

4.3 Testovacie prostredie

/* * testovacie prostredie * spôsoby testovania * použité nástroje - python3 - dig */

Literatúra

[1] *Linux: is there a read or recv from socket with timeout?* 2021. [Online; zmenené 1. január 2021]. Dostupné z: https://stackoverflow.com/questions/2876024/linux-is-there-a-read-or-recv-from-socket-with-timeout.

[2] *Smerovanie*. 2023. [Online; zmenené 19. január 2023]. Dostupné z: https://sk.wikipedia.org/wiki/Smerovanie.

[3] Matoušek, P. *Pokročilé programování síťovych aplikací TCP/IP*. 2014. Dostupné z: https://moodle.vut.cz/pluginfile.php/707796/mod_resource/content/4/isa-sockets.pdf.

[4] Matoušek, P. *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1. Dostupné z: https://www.fit.vut.cz/research/publication/10567.

[5] Mockapetris, P. *Domain names - implementation and specification* [RFC 1035]. RFC Editor, november 1987. DOI: 10.17487/RFC1035. Dostupné z: https://www.rfc-editor.org/info/rfc1035.

[6] Semwal, S. D. *C Program to display hostname and IP address*. 2023. [Online; zmenené 18 apríl 2023]. Dostupné z: https://www.geeksforgeeks.org/c-program-display-hostname-ip-address/.