VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta Informačních Technologií

Dokumentace k projektu pro předmět ISA

DNS Resolver

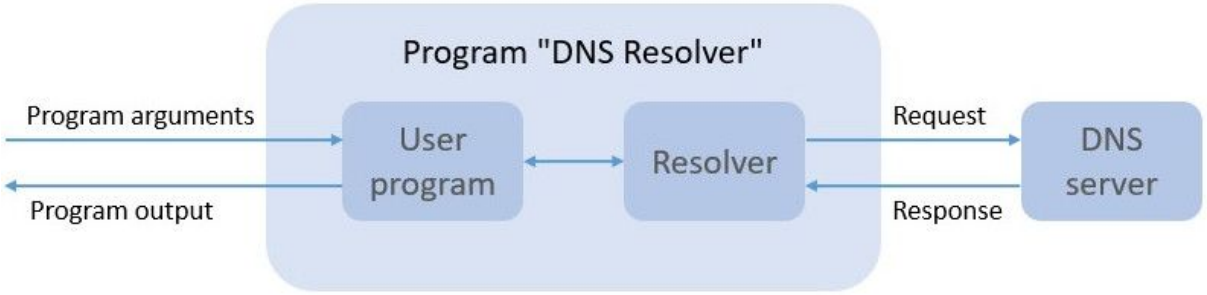
18. listopadu 2019

Mark Menzynski, xmenzy00

Obsah

Úvod do problematiky

Tento manuál byl vytvořen jako dokumentace k projektu “DNS Resolver” k předmětu Síťové aplikace a správa sítí. Popisuje základní informace o programu, návod na použití a také slouží pro popis zajímavých pasáží při implementaci řešení.
DNS Resolver je program nebo služba, která převádí mezi doménovými jmény a IP adresami. Běžně přijímá dotazy, které pak rozesílá DNS serverům po síti. V tomto případě se jedná o program na straně klienta, který zprostředkovává rozhraní mezi uživatelem a serverem, který samotné převádění provádí.



Program může být použit na zjištění IP adresy daného doménového jména, doménového jména dané IP adresy, doplňujících informací k záznamu pro překlad DNS apod.

Popis a spuštění programu

Program je určen pro operační systémy GNU/Linux. Před použitím je potřeba zkompilovat program příkazem **`make`** v adresáři programu.
Program se obsluhuje přes terminál a lze ho spustit následovně:

`dns [-r] [-x] [-6] -s server [-p port] adresa`

Kde:

`dns`	Název zkompilovaného programu.
`-h`	Nápověda.
`-r` `(rekurzivní)`	Požadovat rekurzi.
`-6` `(IP verze 6)`	Dotaz typu AAAA.
`-s` `(server)`	IP adresa nebo doménové jméno serveru, kam zaslat dotaz.
`-p` `(port)`	Port, na který se má zaslat dotaz (výchozí 53).
`adresa`	Adresa, na kterou se tázat.

Program zasílá dotaz zvolenémů serveru a dpověď serveru srozumitelně vypíše na výstup programu. Při chybě program vypisuje informace o chybách na chybový výstup, ten je možný utlumit připsáním **`2>/dev/null`** k argumentům programu.

Implementace programu

Program je napsán v programovacím jazyce C s pomocí běžných systémových knihoven. A to následujících:

- ❏ `stdio.h`
- ❏ `stdlib.h`
- ❏ `string.h`
- ❏ `unistd.h`
- ❏ `errno.h`
- ❏ `stdbool.h`
- ❏ `sys/socket.h`
- ❏ `netinet/in.h`
- ❏ `arpa/inet.h`
- ❏ `netdb.h`

Zdrojový kód se nachází ve 2 následujících souborech:

- ➢ `dns-resolver.c`
- ➢ `dns-resolver.h`

Hlavní funkce `main()` rozčleňuje program na následující podčásti:

1. Parsování argumentů
2. Vytvoření socketu
3. Navázání spojení se serverem
4. Inicializace datagramu
5. Vygenerování ID pro identifikaci datagramu
6. Přidání DNS hlavičky do datagramu
7. Přidání dat dotazu do datagramu
8. Zaslání datagramu
9. Příjem odpovědi serveru a uložení do bufferu
10. Rozepsání základních informací zřejmých z DNS hlavičky odpovědi
11. Vypsání původního dotazu (Question)
12. Vypsání odpovědi (Answer)
13. Vypsání autoritativních informací (Authority)
14. Vypsání dodatečných informací (Additional)
15. Konec programu

Argumenty jsou zparsovány pomocí funkce **getopt()** z knihovny **unistd.h.** Zde se ze vstupu načtou všechny potřebné informace viz. Popis a spuštění programu. Pro vytvoření socketu a navázání spojení jsou použity funkce socket a connect. Funkce connect slouží pro tzv. Spojovanou Schránku UDP, ten zajišťuje například zjištění dostupnosti serveru na síti.
Struktura DNS datagramu:

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Pro datagram je použita struktura **buffer**, jeho definice je následující:

```
struct buffer {
      char *data;
      uint32_t pos;
};
```

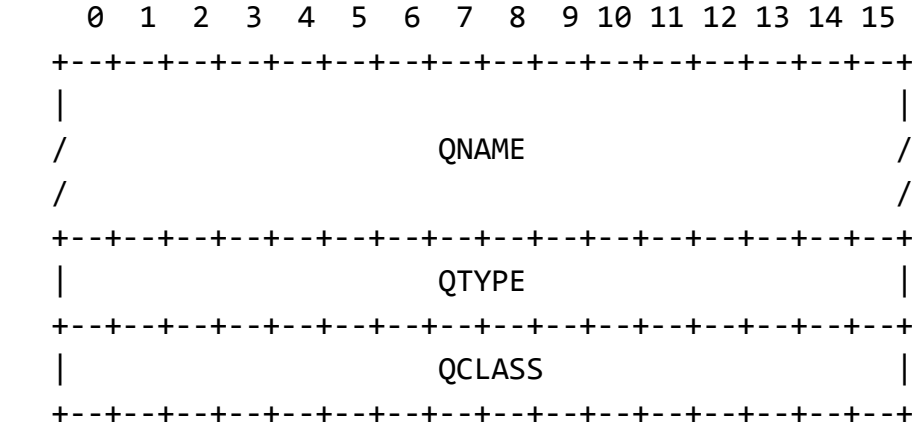
Data jsou při inicializaci alokované na velikost 255 Bytů a **pos** je nastavena na hodnotu 0. **Pos** značí aktuální pozici v bufferu, tímto funkce ví na jakou pozici zapisovat nebo číst data.
Struktura
Hlavička je ve funkci **add_dns_header()** naplněna podle následující struktury:

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Struktura je v kódu definována následujícím způsobem, přičemž pořadí některých položek se může lišit podle architektury:

```
struct dns_header {
      uint16_t    id :16;      /* identification number */
      uint16_t    qr :1;       /* query or response */
      uint16_t    opcode :4;   /* type of query */
      uint16_t    aa :1;       /* authoritative answer */
      uint16_t    tc :1;       /* truncation */
      uint16_t    rd :1;       /* recursion desired */
      uint16_t    ra :1;       /* recursion available */
      uint16_t    unused :1;
      uint16_t    ad :1;       /* authentic data */
      uint16_t    cd :1;       /* checking disabled */
      uint16_t    rcode :4;    /* response code */
      uint16_t    qdcount :16;    /* number of question entries */
      uint16_t    ancount :16;    /* number of answer entries */
      uint16_t    nscount :16;    /* number of authority entries */
      uint16_t    arcount :16;    /* number of resource entries */
};
```

V této funkci bývá většina položek ponechána v nule. Položka **id** je zvoleno podle PID běžícího procesu. Položka **rd** se mění podle vstupního argumentu **-r** a **qdcount** je vždy nastaven na 1 značící 1 dotaz v datagramu.
Data dotazu jsou přidány ve funkci add question. Jeho struktura je následující:



Kde:

QNAME značí doménové jméno, jehož adresu chceme získat. Je ale zadáno v jiném formátu, kde například doménové jméno www.vutbr.cz by bylo zapsáno jako `/003www/005vutbr/002cz/000`. Tento převod provádí funkce **`encode_hostname();`**
QTYPE označuje typ odpovědi, kterou chceme získat. To bývá dotaz na **A** nebo **AAAA** záznam. V případě reverzního dotazu je typ **PTR** (pointer).
QCLASS označuje třídu. Ta je v tomto případě vždy **IN** (internet).

Zaslání datagramu provádí funkce **`send()`**. Příjem odpovědi provádí funkce **`recv()`**. Zde hrozí, že když žádná odpověď nepřijde, program čeká nekonečně dlouho dobu, a proto je před navázáním spojení nastaven timeout na 5 sekund pomocí funkce **`setsockopt()`**.
Vypsání základních informací jako Authoritative, Recursive, Truncated je jednoduše pomocí flagů v hlavičce, a to: **aa**, **rd && ra** a **tc**. Pro vypsání dotazu slouží funkce **`print_questions(),`** pro vypsání odpovědi, autoritativních a dodatečných informací funkce **`print_resource()`**.

Přehled zajímavějších pasáží z kódu

Dekomprese při převodu doménového jména síťového formátu na klasický čitelný formát

```
    next_length = name[next_length];
    if (isPointer(next_length)) {
        /* If 2 most significant bits are both 1, it is a pointer, not a length
         * and next 14 bits are offset */
        uint16_t offset = pointer_to_offset(&name[pos]); // Calculate and offset
        char * pointed_name = strdup(&buff->data[offset]); // Copy the content from offset
        char * decompressed_name = decode_name(buff, pointed_name); // Decode the content
        int16_t additional_len = strlen(decompressed_name);
        hostname = realloc(hostname, name_length + additional_len); // Reallocate the string

        memcpy(&hostname[pos], decompressed_name, additional_len + 1); // Copy the decoded
content to string
        name_length += additional_len; // Update the length
        break;
    }
```

Detekce komprese je implementována ve funkci **`isPointer()`**. Detekce probíhá tak, že se zkontrolují první 2 bity ve znaku. Pokud oba bity jsou v 1, pak se jedná o pointer. Offset pak odkazuje na jinou pozici v datagramu, kde se nachází pokračování doménového jména. Toto doménové jméno již bylo použito předtím, a tak se jedná o “recyklování” již použitých jmen v datagramu. Toto je implementováno uvnitř funkce **`decode_hostname().`**

Reverzace IPv6 adresy

```
/* Inverse the address and save it into qname */
if (inet_pton(AF_INET6, address, &addr) == 1) {
    qname = malloc(qname6_length);
    uint8_t *addr8 = addr.s6_addr;
    char c[4];
    int iter = 0;

    /* Each byte contains 2 address characters each in 4 bits, this reads them and also directly adds
     * the lengths for network hostname format */
    for (int i = 15; i >= 0; i--) {
        c[0] = 1;
        c[1] = hex_to_char(addr8[i] & 0x0f);
        c[2] = 1;
        c[3] = hex_to_char(addr8[i] >> 4);
        memcpy(&qname[iter], c, 4); // Copy the parsed byte into string
        iter += 4;
    }
    memcpy(&qname[iter], encode_hostname("ip6.arpa"), 9); // Add suffix
```

Funkce inet_pton převede řetězcový formát na binární. Pokud návratová hodnota není 1, pak se nejedná o validní IPv6 adresu. Binární formát ale není také ideální. Každý byte obsahuje 2 znaky, každý o velikosti 4 bitů. Při procházení adresy rovnou dochází i k zakódování na síťový formát. Proto se před každým znakem ukládá 1 značící 1 znak. Na konci je taky přidána koncovka značící reverzní IPv6 adresu.

Literatura

- RFC 1035 https://tools.ietf.org/html/rfc1035
- RFC 3596 https://tools.ietf.org/html/rfc3596
- Linux manuálové stránky přístupné příkazem **`man`** (nebo https://linux.die.net/man/)