ISA – protokol k projektu DNS resolver

Jakub Pomsár
xpomsa00

16. listopadu 2023

Obsah

1 Úvod do problematiky **3**
1.1 Úvod do resolveru . . . 3
1.2 Zjednodušený resolver (Stub resolver) . . . 3

2 Návrh **3**

3 Popis implementace **3**
3.1 Obecný popis implementace . . . 3
3.2 Podrobnější popis zajímavých částí . . . 4
3.2.1 Překlad adresy DNS serveru . . . 4
3.2.2 Kontrola adres . . . 4
3.2.3 Kontrola obdržených zpráv . . . 5
3.2.4 Reverzní adresa . . . 5

4 Návod **6**

5 Testování **6**
5.1 Automatické testy . . . 6
5.2 Testovací prostředí . . . 7
5.3 Specifikace lokálního zařízení . . . 7

1 Úvod do problematiky

1.1 Úvod do resolveru

Resolver je program, který slouží k propojení uživatelských programů s doménovými jmennými servery. V nejjednodušším případě resolver obdrží požadavek od uživatelského programu ve formě podprogramového volání, systémového volání atd. a vrátí požadované informace ve formě kompatibilní s datovými formáty místního hosta. [3] Resolver je umístěn na stejném stroji jako program, který vyžaduje služby resolveru, ale může být nutné konzultovat jmenné servery na jiných hostech. Protože resolver může potřebovat konzultovat několik jmenných serverů nebo může mít požadované informace v místní mezipaměti, doba, kterou resolver potřebuje k dokončení, se může značně lišit, od milisekund po několik sekund. [3] Prvním krokem, který resolver provede, je transformace požadavku klienta, do vyhledávací specifikace pro záznamy zóny na konkrétním jménu, které odpovídají konkrétnímu typu otázky (`QTYPE`) a třídě otázky (`QCLASS`). [4]

1.2 Zjednodušený resolver (Stub resolver)

Jedna možnost implementace resolveru spočívá v přesunu rezoluce z místního stroje do DNS serveru, který podporuje rekurzivní dotazy. To může poskytnout snadný způsob poskytování doménové služby na počítači, který nemá zdroje k provedení funkce resolveru, nebo může centralizovat mezipaměť pro celou místní síť nebo organizaci. Jediné, co zbylý nástroj potřebuje, je seznam adres názvových serverů, které provedou rekurzivní požadavky. Tento typ resolveru pravděpodobně potřebuje informace v konfiguračním souboru, protože pravděpodobně nemá sofistikovanost na to, aby je lokalizoval v databázi domény. Uživatel také potřebuje ověřit, zda uvedené servery provedou rekurzivní službu; názvový server může odmítnout provedení rekurzivních služeb pro kteréhokoli nebo všechny klienty. Uživatel by měl konzultovat místního správce systému, aby našel názvové servery ochotné provádět tuto službu. [3]

2 Návrh

Po spojení informací z zadání, RFC 1034 a RFC 1035 je resolver implementován jako "stub resolver", který nemá seznam DNS serverů, které může kontaktovat při dotazech. Na rozdíl od *stub resolveru* z RFC, má pouze adresu jednoho serveru, která je zadána uživatelem. Podle standardu může resolver v některých případech komunikovat pomocí protokolu `TCP`, ale dle zadání to není požadováno [3]. Z důvodu zjednodušení implementace probíhá komunikace výhradně přes `UDP`. `TCP` komunikace a vyhledávání v mezipaměti tedy není implementováno. Resolver tedy konzultuje dotazy pouze s jedním DNS serverem a interpretuje jeho odpovědi a chybové stavy. Resolver dle zadání musí podporovat následující typy DNS záznamů: *A, AAAA, CNAME, PTR*. Kromě toho jsou rovněž podporovány záznamy typu *NS (Name Server)* a *SOA (Start of Authority)*, které se často vyskytují při nerekurzivních dotazech.

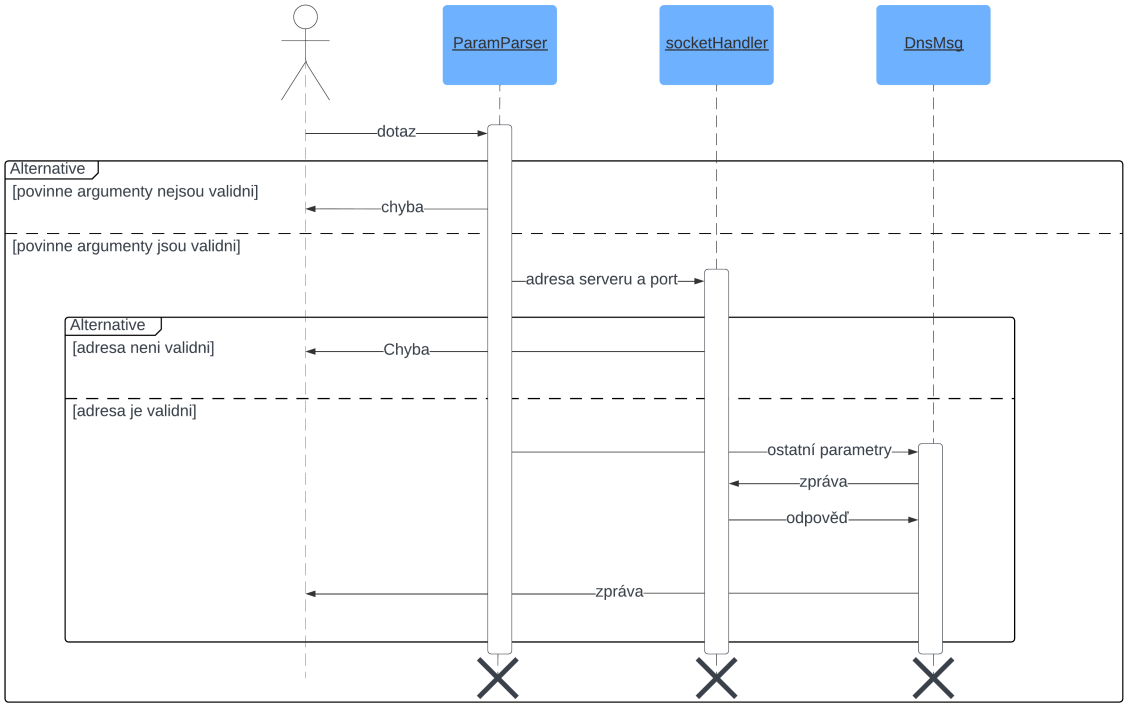
3 Popis implementace

3.1 Obecný popis implementace

Dns resolver je implementován pomocí 3 tříd: `SocketHandler`, `ParamParser` a `DnsMsg`. Třída `ParamParser` obsahuje třídní metodu `paramProcess`, která zpracovává argumenty přijaté při spuštění programu. Následně předá zpracovaný port a adresu serveru objektu `SocketHandler`, který otevře UDP socket pro příslušnou adresu. V případě potřeby si vyhledá IP adresu pomocí funkce `gethostbyname()`.

Dále objekt `ParamParser` předá zpracované argumenty objektu `DnsMsg`, který příslušně nastaví hlavičku a otázku DNS zprávy. Předá vytvořenou zprávu objektu `SocketHandler`, který ji zašle na požadovaný DNS server a očekává odpověď. Následně jí vrátí objektu `DnsMsg`. Ten zpracuje přijatou

zprávu a vypíše ji uživateli na standardní výstup. V případě chyby resolveru nebo serveru na chybový výstup. Nakonec se program ukončí s odpovídajícím návratovým kódem.



Obrázek 1: sekvenční diagram komunikace mezi objekty

3.2 Podrobnější popis zajímavých částí

3.2.1 Překlad adresy DNS serveru

V případě, že resolver neobdrží IP adresu DNS serveru, ale pouze doménové jméno DNS serveru, je nutné toto jméno přeložit na odpovídající IP adresu. Resolver tedy musí provést dotaz na překlad doménového jména. Pro tento případ je použita funkce `gethostbyname()`, která může vrátit několik IP adres. Resolver následně iteruje přes vrácené IP adresy a vytváří schránku, při prvním úspěchu, když najde platnou IP adresu, nebo je ukončen s chybovým návratovým kódem. Existuje také možnost rozšíření implementace nad rámec zadání, jak resolver dokáže zjistit IP adresu bez použití funkce `gethostbyname()`, která ovšem není implementována.

V tomto případě může resolver nahlédnout do souboru `/etc/hosts` , zda se v něm nachází záznam o překladu doménového jména. V případě neúspěchu nahlédne do souboru `/etc/resolv.conf`, kde jsou implicitně zadány servery, kterých se může dotazovat. Tímto způsobem může resolver získat potřebnou IP adresu. Celkově tedy provede svou funkci dvakrát – nejprve pro získání IP adresy DNS serveru a poté pro dotaz na adresu zadávanou uživatelem.

3.2.2 Kontrola adres

Ve stávající implementaci není provedena žádná kontrola získaných adres od uživatele, a předpokládá se, že uživatel zadá adresu ve správném formátu odpovídajícím použitým symbolům či jejich kombinacím. V případě, když se uživatel dotazuje na reverzní dotaz s IPv6 adresou, resolver požaduje její celý nezkrácený formát. Tím se zaručuje správný převod z původní adresy na reverzní. Chybný vstup bude odhalen až při dotazu na DNS server, který vrátí odpovídající chybové hlášení. Resolver následně

propaguje chybu a vrátí totožný návratový kód, jaký obdržel od DNS serveru. Přijímání zkrácených IPv6 adres může být považováno za možné rozšíření do budoucna. V následujícím příkladu je spuštěn DNS resolver s adresou `1.1.1.1`, avšak není uveden příznak `-x`, který nastavuje typ dotazu na reverzní dotaz. Adresa je tedy špatně přeložena, a DNS server zasílá chybu 3 `Name Error [No such name]`, na základě které resolver ukončuje program s návratovým kódem 3.

```
$ ./dns -s dns.google 1.1.1.1
Authoritative: No, Recursive: No, Truncated: No, ERROR: Name Error [No such name]
Question section (1)
  1.1.1.1., A, IN
Answer section (0)
Authority section (1)
  , SOA, IN, 84776, a.root-servers.net., nstld.verisign-grs.com.,
   2023111500, 1800, 900, 604800, 86400
Additional section (0)

$ echo $?
3
```

3.2.3 Kontrola obdržených zpráv

Je prováděna kontrola jednotlivých příznaků hlavičky příchozí zprávy. Zvláště je kontrolováno ID, aby se zajistilo, že se shoduje ID dotazu s odpovědí. Také se kontroluje příznak `QR`, který určuje, zda přijímaná zpráva je ve skutečnosti odpovědí, a ne další dotaz. V případě, že je zpráva zkrácena, musí být nastaven příznak `TC` DNS serverem. Resolver by měl zahodit UDP datagram a přejít na TCP [1]. Ovšem komunikace pomocí TCP není implementována, a v tomto případě se vypíše celá, i když zkrácená, zpráva spolu s informací `Truncation : Yes`.

3.2.4 Reverzní adresa

Standard RFC 1035 specifikuje způsob zasílání reverzních dotazů pomocí nastavení *OPCODE* v hlavičce zprávy na hodnotu 2, což reprezentuje reverzní dotaz [4]. V praxi většina serverů odpoví chybovou zprávou `not-implemented error`, což je povinná vlastnost implementace DNS serveru. Alternativou je zaslat IP adresu v reverzním formátu s příponou `in-addr.arpa` pro IPv4 nebo v reverzním formátu s příponou `ip6.arpa` pro IPv6 adresy[4][2].

4 Návod

Program lze spustit následovně:

USAGE:
`./dns [-r] [-x] [-6] [-p port] -s server adresa`

Kde:

- Symbol **r** označuje požadovanou rekurzi (*recursion desired*).
- Symbol **x** signalizuje dotaz na reverzní adresu (*záznam PTR*).
- Symbol **6** indikuje dotaz na IPv6 adresu (*záznam AAAA*).
- Symbol **p** s argumentem **port** umožňuje specifikovat jiný port než výchozí *53*.
- Povinný symbol **s** s argumentem **server** definuje DNS server, na který je dotaz odeslán.
- Povinný argument **adresa** určuje předmět dotazu.

Kombinaci symbolů **x** a **6** program interpretuje jako reverzní dotaz na IPv6 adresu.

5 Testování

5.1 Automatické testy

V rámci implementace resolveru byly vytvořeny také automatické testy, které ověřují správné zpracování argumentů, výpis jednotlivých sekcí (*answer, authority a additional*) a nastavení zjištěných příznaků z hlavičky (*Authority, Recursive*). Tyto testy spoléhají na interpret Python3 a program dig pro provádění ověření. Tímto způsobem jsou zajištěny klíčové aspekty správné funkcionality resolveru.

Testy lze spustit příkazem:

make `test`

Výsledky testu jsou vygenerovány ve složce `test-artifacts` s následující hierarchií:

- AAAA-record/
- Additional/
- A-record/
- args/
- args/
- Authority/
- misc/
- reverse-AAAA-record/
- reverse-A-record/

Příklad jednoho výsledku testu:

```
$ cat test-artifacts/A-record/nes.fit.vutbr.cz
TEST A-record for nes.fit.vutbr.cz PASSED
DIG output:
dig @8.8.8.8 +noall +answer A nes.fit.vutbr.cz
{'name': 'nes.fit.vutbr.cz.', 'class': 'IN', 'type': 'A', 'ttl': '14400',
'data': '147.229.8.16'}

DNS output:
./dns -r -s 8.8.8.8 nes.fit.vutbr.cz
{'name': 'nes.fit.vutbr.cz.', 'class': 'IN', 'type': 'A', 'ttl': '13755',
'data': '147.229.8.16'}
```

5.2 Testovací prostředí

Projekt byl testován na několika testovacích prostředích:

- server merlin.fit.vutbr.cz
- server eva.fit.vutbr.cz
- lokální zařízení

5.3 Specifikace lokálního zařízení

```
OS: EndeavourOS x86_64
Host: Modern 14 B5M (REV:1.0)
Kernel: 6.1.62-1-lts
CPU: AMD Ryzen 7 5700U (16) @ 4.37 GHz
GPU: AMD Lucienne

Python 3.11.5

DiG 9.18.19
```

Reference

[1] Robert Elz and Randy Bush. *Clarifications to the DNS Specification*, July 1997. `http://www.rfc-editor.org/rfc/rfc2181.txt`.

[2] Vladimir Ksinant, Christian Huitema, Dr. Susan Thomson, and Mohsen Souissi. *DNS Extensions to Support IP Version 6*, November 1987. `http://www.rfc-editor.org/rfc/rfc3596.txt`.

[3] P. Mockapetris. *Domain names - concepts and facilities*, November 1987. `http://www.rfc-editor.org/rfc/rfc1034.txt`.

[4] P. Mockapetris. *Domain names - implementation and specification*, November 1987. `http://www.rfc-editor.org/rfc/rfc1035.txt`.