VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí – dokumentace projektu

Filtrující DNS resolver

10. listopadu 2020

Dominik Nejedlý

Obsah

1 Úvod **1**

2 Návrh a popis implementace **1**
2.1 Modul `charPList.c` 1
2.2 Modul `dns.c` 1
2.2.1 Zpracování vstupních argumentů 1
2.2.2 Komunikace s klientem 1
2.2.3 analýza DNS paketu 2
2.2.4 Způsob filtrování blokovaných domén 2
2.2.5 Rezoluce DNS dotazu 2
2.2.6 Chybové odpovědi 3
2.2.7 Uvolňování paměti a uzavírání soketů při přerušení 3

3 Návod na použití **4**

4 Návratové hodnoty programu **4**

5 Závěr **4**

1 Úvod

Cílem projektu bylo vytvořit program, který filtruje dns dotazy typu A směřující na domény, které se nachází v dodaných souborech, a jejich poddomény. Na nežádoucí dotazy, případně při selhání zpracování dotazu, program pošle klientovi zprávu s odpovídajícím chybovým kódem viz 2.2.6. Ostatní dotazy jsou dále přeposílány specifikovanému DNS resolveru, jehož odpovědi jsou navráceny původním tazatelům.

Program funguje jako UDP server, tedy přijímá dotazy od jakéhokoli klienta, a při svém běhu nevypisuje žádné informace. Pokud však nastane problém znemožňující jeho další běh, program je ukončen s daným chybovým kódem viz 4 a na standardní chybový výstup je vytištěna zpráva blíže specifikující důvod přerušení běhu a ukončení programu. V opačném případě program běží, dokud není zastaven nějakým vnějším signálem.

2 Návrh a popis implementace

Program je napsán v jazyce C a je rozdělen do modulů `dns.c` a `charPList.c` z nichž každý je dále dělen do funkcí.

2.1 Modul `charPList.c`

V tomto modulu jsou definovány funkce pro ovládání jednosměrně vázaného seznamu ukazatelů na znak, přičemž struktury, které ho představují jsou definovány v hlavičkovém souboru tohoto modulu (`charPList.h`). Tento seznam je poté v programu využíván pro uchovávání jmen všech filtrovacích souborů, celý program není tedy omezen pouze na práci s jedním filtrovacím souborem, ale je možné pracovat i s několika filtrovacími soubory současně.

2.2 Modul `dns.c`

Modul obsahuje hlavní tělo programu `main` a funkce obstarávající dílčí úkony nutné pro správný běh programu. V tomto oddíle jsou dále blíže popsány důležité a zajímavé části implementace tohoto modulu a celkové chování programu.

2.2.1 Zpracování vstupních argumentů

Zpracování vstupních argumentů je řešeno pomocí funkce `parseArgs`. Jednotlivé vstupní argumenty jsou procházeny a pokud je nalezen známý přepínač, je načítána hodnota, kterou specifikuje. Samotný přepínač a načítaná hodnota mohou být dva vstupní argumenty za sebou, ale program je zpracuje správně, i pokud se jedná o jeden argument (tedy přepínač a hodnota jsou zadány bezprostředně za sebou bez bílého znaku). Při nalezení neznámého přepínače, chybějící hodnotě za přepínačem známým, chybějícím povinném vstupním argumentu, nevalidní adrese DNS serveru (resolveru), nebo jiné chybě při zpracování argumentů je funkce ukončena a celý program končí s odpovídajícím návratovým chybovým kódem. Pokud je některý přepínač zadán vícekrát, hodnota se kterou se dále pracuje, je poslední nalezená. Tohle ovšem neplatí pro přepínač `-f` (program dokáže filtrovat i přes několik filtrovacích souborů) viz 3.

Pro ověření validní IP adresy, případně pro její získání z doménového jména, je argument představující DNS resolver ihned po jeho načtění poslán do funkce `getaddrinfo` [8]. Tato funkce je použita místo `gethostbyname` [16], jelikož je schopná zpracovat adresy IPv4 i IPv6.

2.2.2 Komunikace s klientem

Aby bylo možné komunikovat s klienty z adres IPv4 i IPv6, jsou pro uchování adresy klienta i tohoto programu použity structury `sockaddr_in6` [14], které umožňují uchovat oba typy adres. Samotná komunikace klienta a tohoto programu poté probíhá přes soket navázaný na tento server pomocí funkce `bind` [7]. Příjem dotazu od klienta zajišťuje ve smyčce volaná funkce `recvfrom` [10] a odeslání odpovědi funkce `sendto` [13].

2.2.3 analýza DNS paketu

V hlavičce každého přijatého DNS paketu od klienta musí být nastaveno, že se jedná o dotaz (`QR` bit je nastaven na 0), hodnota rezervovaných bitů by měla být dle standardu nulová (tedy `Z` bit/bity jsou nastaveny na 0). Zde dochází k mírnému rozporu mezi dokumentací DNS a odchycenou komunikací v programu wireshark viz [1], jelikož dokumentace uvádí, že rezervované bity ve `FLAGS` jsou 3 viz [17], ovšem v odchycené komunikaci, je tak nahlíženo pouze na 1. V tomto programu je dle odchycené komunikace kontrolován pouze 1 rezervovaný bit. Dále je v hlavičce paketu kontrolován počet otázek (podporována pouze 1). V sekci otázky je kontrolován její typ (podporovány pouze dotazy typu A).

Pokud jakákoli z výše uvedených kontrol selže, je klientovi odeslána daná chybová odpověď viz 2.2.6.

2.2.4 Způsob filtrování blokovaných domén

Nejprve je nutné z paketu získat dotazované doménové jméno, což řeší funkce `getDomainName`. Doménové jméno je v paketu uloženo jako sekvence značek, kde spodních 6 bitů prvního oktetu značky určuje počet jejích následných znaků. Konec učuje značka v níž je počet znaků nulový. Funkce tedy načítá postupně značky, dokud na právě zmíněnou nenarazí nebo dokud není dosažena maximální možná délka doménového jména (253 znaků).

Filtrování probíhá postupným průchodem přes všechny fitrovací soubory v cyklu. Každý soubor je nejprve otevřen a poté je po řádcích načítán. Soubory je tedy možné upravovat i za běhu programu a jelikož obsahy souborů nejsou nahrávány do paměti, je zajištěna i nižší paměťová spotřeba programu.

Prázdné řádky, řádky obsahující pouze bílé znaky a řádky jejichž prvním nebílým znakem je '#' jsou přeskakovány. Ostatní řádky jsou upraveny tak, aby bylo načteno pouze filtrované doménové jméno. Ze začátku a konce řádku jsou odstraněny případné bílé znaky. Pokud se na řádku nachází více znakových sekvencí oddělených bílými znaky, je načtena pouze první a zbytek řádku je přeskočen atd. Program je tedy schopný akceptovat i hůře zformátovaný filtrovací sobor, pokud je zachována alespoň nějaká konzistence.

Poté, co je z filtrovacího souboru načtena blokovaná doména a je upraven její formát, je tato doména vyhledávána v dotazovaném doménovém jméně pomocí funkce `strstr` [19].

Pokud je nalezena shoda, blokovaná doména je podřetězcem dotazovaného doménového jména a funkce vrátí ukazatel na počáteční místo shody v dotazovaném doménovém jméně. Ukazatel tak nyní představuje podřetězec dotazovaného doménového jména, přičemž ale tento celý podřetězec nemusí být roven blokované doméně, jelikož se za touto nalezenou shodou mohou nalézat další znaky. Z toho důvodu se tedy dále použije funkce `strcmp` [18] k porovnání podřetězce určeného získaným ukazatelem a blokované domény. Pokud jsou oba řetězce totožné, znamená to, že dotazované doménové jméno je onou blokovanou doménou, nebo její poddoménou. Cyklus končí a klientovi je odeslána odpověď informující o odmítnutí zpracování dotazu viz 2.2.6.

Pokud se ale řetězce liší, celý cyklus vyhledávání podřetězce se opakuje, ovšem až od pozice za poslední nalezenou shodou. Celý cyklus končí, pokud blokovaná doména není v dotazovaném doménovém jméně nalezena, což znamená, že doména není blokovaná a je možné tento dotaz přeposlat k rezoluci.

2.2.5 Rezoluce DNS dotazu

Překlad doménového jména na IP adresu řeší vstupním parametrem daný DNS resolver. Tomu je ale nutné přeposlat paket určený ke zpracování.

Všechny adresy daného DNS resolveru jsou získány funkcí `getaddrinfo` [8]. Ta vrací list struktur `addrinfo`, z nichž každá obsahuje právě jednu možnou adresu daného resolveru. Tento list je procházen v cyklu. Vždy se nejprve vytvoří soket typu odpovídajícího dané adrese a poté se pomocí funkce `connect` [5] pokusí připojit k danému serveru. Způsob připojení je řešen spojovnými schránkami UDP. Tento způsob byl zvolen, jelikož program během svého běhu přeposílá dotazy k rezoluci vždy pouze jednomu danému DNS resolveru, je tedy možné hned při spuštění ustavit tohle spojení a zjistit ještě před zahájením komunikace, zdali je server vůbec dostupný, což je výhoda oproti klasickému způsobu UDP.

V případě úspěšného navázání spojení je cyklus ukončen. Přeposlání paketu k rezoluci zajišťuje funkce `send` [12] a příjem odpovědi funkce `recv` [9].

Pokud ale v průběhu cyklu došlo k chybě (nepodařilo se vytvořit soket, či navázat spojení), je celý proces opakován, dokud není spojení navázano s nějakou další možnou adresou daného serveru, nebo dokud již není žádná další možná adresa, což znamená, že není možné zřídit komunikační kanál a celý program je ukončen s odpovídající chybovou návratovou hodnotou viz 4.

2.2.6 Chybové odpovědi

V případě špatného formátu DNS paketu dotazu od klienta, dotazu na blokovanou doménu, interní chyby programu při zpracování paketu, nepodporovaného typu dotazu nebo více než jednoho doménového jména v jednom paketu k rezoluci je klientovi odeslána chybová odpověd, která ho o nastalém problému informuje.

Formování této odpovědi má na starosti funkce `setErrorResponse`. Ta nejprve nastaví v hlavičce paketu, že se jedná o odpověď (`QR` bit nastaví na 1). Pokud byla v dotazovaném paketu vyžadováná rekurze v rezoluci (`RD` bit nastaven na 1), je v odpovědi nastaveno, že rekurze je dostupná (`RA` bit nastaven na 1), aby se předešlo nekonzistenci.

Dále je v hlavičce paketu nastaven chybový kód (`RCODE`, jedná se o 4 bity reprezentující číselnou hodnotu), podle typu zaznamenaného problému:

- 1 Špatný formát DNS paketu dotazu (`QR` bit nebo `Z` bit nastaven na 1)
- 2 Chyba v programu během analýzy a zpracování paketu
- 4 Nepodporovaný typ dotazu nebo paket s více než jednou doménou k rezoluci
- 5 Dotaz na blokovanou doménu

V případě, že počet otázek v paketu je roven 1, je zbytek hlavičky DNS paketu za počtem otázek vynulován a celou odpověď tak tedy tvoří pouze hlavička DNS paketu a sekce otázky. Pokud se ale v paketu nachází otázek více (velmi nepravděpodobné, globálně nepodporováno), jsou provedena nastavení chybové odpovědi, ale dále již zbytek hlavičky za počtem otázek není nulován, z důvodu vyšší režie v programu.

2.2.7 Uvolňování paměti a uzavírání soketů při přerušení

Jelikož hlaní tělo programu tvoří nekonečná smyčka, je nutné jej ukončit nějakým vnějším signálem (např. `SIGINT`). Při jeho přijetí ale program okamžitě skončí a neuvolní alokované zdroje ani neuzavře používané sokety. Aby bylo možné tyto operace provést, je nutné nastavit speciální obsluhu daného signálu. V tomto programu se ke zpracování signálu přerušení používá `sigIntHandler`, který namísto okamžitého přerušení programu, ovládá globální proměnnou, která celý nekonečný cyklus v hlavním těle programu řídí, přičemž cyklus při následném testování podmínky skončí.

V cyklu se ale používají blokující funkce `recvfrom` [10] a `recv` [9], ve kterých může celý cyklus uváznout, dokud tyto funkce neproběhnou (tedy nepřijmou nějakou zprávu). Může se tedy stát, že má být program ukončen, ale jelikož je pozměněna funkce signálu přerušení, tak k ukončení nedojde, dokud nebude v dalším běhu cyklu testována podmínka.

Tomuto uváznutí je možné předejít nastavením časového limitu pro příjem funkcí `select` [11]. Podmínka na ukončení cyklu je tedy pravidelně po určitém limitu testováná a navíc se zabrání i uváznutí v čekání na odpověď od DNS resolveru. Tato funkce také sama reaguje na signál přerušení, pokud je zrovna aktivní a okamžitě skončí, což odstřaňuje i prodlevu mezi následným testováním podmínky nekonečné smyčky.

Je zde ale jeden problém. Pokud je přerušení přijato po testování podmínky cyklu, ale před vstupem do této funkce, program nekončí, ale čeká požadovaný časový limit na příjem zprávy, jež bude i zpracována, pokud přijde.

Aby bylo možné předejít této běhové chybě, je nutné použít funci `pselect` [6] a před vstupem do onoho nekonečného cyklu v hlavním těle programu nastavit blokování signálu přerušení funkcí `sigprocmask` [15]. Funkce `pselect` [6] má v zásadě stejnou funkčnost jako funkce `select` [11], ovšem dovoluje uvolnit zablokovaný signál během svého vykonávání, proto s jejím využitím nemůže vzniknout dříve popsaná běhová chyba a reakce na přerušení je tak tedy vždy okamžitá.

Po vystoupení z nekonečné smyčky v hlavním těle programu při obdržení signálu přerušení se uvolní alokované zdroje a uzavřou sokety. Následně program odblokuje signál přerušení a sám se jím ukončí, aby byla zachována konzistence a program skončil daným signálem, jak je obecně očekáváno viz [2].

3 Návod na použití

Program se spouští následujícím příkazem:

```
./dns -s server [-p port] -f filter_file
```

Parametry lze zadávat i bez dělícího bílého znaku:

```
./dns -sserver [-pport] -ffilter_file
```

Oba výše zmíněné způsoby se dají také kombinovat.

Vstupní parametry:

- `-s server` DNS server, na který jsou přeposílány dotazy k rezoluci (povinný)
- `-p port` Číslo portu určeného pro příjem příchozích dotazů (volitelný, výchozí hodnota je 53)
- `-f filter_file` Soubor s nežádoucími doménami (povinný, je možné filtrovat i přes několik souborů)

Pro výpis nápovědy je možné použít parametr `-h` nebo `--help`. V tomto případě program vytiskne nápovědu a úspěšně skončí.

4 Návratové hodnoty programu

Výčet a bližší popis návratových hodnot tohoto programu:

- 0 Úspěšné ukončení programu (např. po výpisu nápovědy)
- 1 Chybějící nebo neplatné vstupní parametry programu
- 2 Vytvoření soketu pro komunikaci mezi tímto programem a klientem se nezdařilo
- 3 Navázání soketu pro komunikaci mezi tímto programem a klientem na adresu a port se nezdařilo
- 4 Vytvoření soketu pro komunikaci mezi tímto programem a DNS resolverem se nezdařilo
- 5 Připojení k danému DNS resolveru se nezdařilo
- 6 Odeslání odpovědi klientovi se nezdařilo
- 99 Interní chyba tohoto programu (např. chyba při alokaci paměti, blokování signálu atd.)

5 Závěr

Při práci na tomto projektu jsem si mohl vyzkoušet práci se sokety a knihovnami používanými v síťovém prostředí, jejichž použití bylo nutné nastudovat. K tomu dobře posloužila jejich dokumentace. Informace o UDP komunikaci a DNS jsem čerpal především z přednášek předmětu ISA. Inspirací při práci mi také byly příklady jednoduchého UDP serveru a klienta, které je možné nalézt v souborech na stránce tohoto předmětu.

Dále bylo nutné nastudovat strukturu DNS paketu podle `RFC 1035` [17], porozumět blokování viz [3] a obsloužení viz [2], [4] systémových signálů a nastavení časového limitu pomocí funkce `pselect` [6]. Během implementace jsem se také zdokonalil v používání programu wireshark viz [1].

Zdroje

[1] Combs, G.: Wireshark. [online], [vid. 2020-11-10]. Dostupné z: https://www.wireshark.org/

[2] Cracauer, M.: Proper handling of SIGINT/SIGQUIT. [online], únor 2013, [vid. 2020-11-10]. Dostupné z: https://www.cons.org/cracauer/sigint.html

[3] Free Software Foundation, I.: 24.7 Blocking Signals. [online], [vid. 2020-11-10]. Dostupné z: https://www.gnu.org/software/libc/manual/html_node/Blocking-Signals.html

[4] Ghosh, B.: How to use signal handlers in C language? [online], [vid. 2020-11-10]. Dostupné z: https://linuxhint.com/signal_handlers_c_programming_language/

[5] Kerrisk, M.: connect(2) — Linux manual page. [online], duben 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/connect.2.html

[6] Kerrisk, M.: The new pselect() system call. [online], květen 2006, [vid. 2020-11-10]. Dostupné z: https://lwn.net/Articles/176911/

[7] Kerrisk, M.: bind(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/bind.2.html

[8] Kerrisk, M.: getaddrinfo(3) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man3/getaddrinfo.3.html

[9] Kerrisk, M.: recv(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/recv.2.html

[10] Kerrisk, M.: recvfrom(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/recvfrom.2.html

[11] Kerrisk, M.: select(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/select.2.html

[12] Kerrisk, M.: send(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/send.2.html

[13] Kerrisk, M.: sendto(2) — Linux manual page. [online], listopad 2020, [vid. 2020-11-10]. Dostupné z: https://www.man7.org/linux/man-pages/man2/sendto.2.html

[14] Kerrisk, M.: ipv6(7) — Linux manual page. [online], září 2017, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man7/ipv6.7.html

[15] Kerrisk, M.: sigprocmask(2) — Linux manual page. [online], září 2017, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man2/sigprocmask.2.html

[16] Kerrisk, M.: gethostbyname(3) — Linux manual page. [online], červen 2020, [vid. 2020-11-10]. Dostupné z: https://man7.org/linux/man-pages/man3/gethostbyname.3.html

[17] Mockapetris, P.: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. [online], listopad 1987, [vid. 2020-11-10]. Dostupné z: https://tools.ietf.org/html/rfc1035

[18] tutorialspoint: C library function - strcmp(). [online], [vid. 2020-11-10]. Dostupné z: https://www.tutorialspoint.com/c_standard_library/c_function_strcmp.htm

[19] tutorialspoint: C library function - strstr(). [online], [vid. 2020-11-10]. Dostupné z: https://www.tutorialspoint.com/c_standard_library/c_function_strstr.htm