**Vysoké učení technické v Brně**

**Fakulta** **informačních** **technologií**



**Síťové aplikace a správa sítí**

**2020/2021**

**Dokumentace projektu**

**Filtrující DNS resolver**

Daniel Pátek (xpatek08) Brno, 2. listopadu 2020

1. Obsah

[1 Úvod 3](#_Toc56261535)

[2 Uvedení do problematiky 3](#_Toc56261536)

[3 Implementace 4](#_Toc56261537)

[3.1 Základní údaje 4](#_Toc56261538)

[3.2 Funkce main() 4](#_Toc56261539)

[3.3 Funkce getDnsRequestData() 5](#_Toc56261540)

[3.4 Funkce isBlacklisted() 5](#_Toc56261541)

[3.5 Funkce getDnsFilter() 6](#_Toc56261542)

[3.6 Funkce clear() 6](#_Toc56261543)

[4 Zdroje 6](#_Toc56261544)

# Úvod

Cílem řešeného projektu je návrh a implementace filtrujícího DNS resolveru. Tento program bude schopný na určeném síťovém portu zachytávat provoz a filtrovat pakety protokolu DNS. Součástí projektu je vytvoření dokumentace.

Program bude podporovat pouze DNS dotazy typu A a bude fungovat na protokolu UDP. Nebude podporovat DNSSEC. Kromě portu, na kterém bude program přijímat dotazy, bude možné specifikovat, na jaký resolver se mají vyfiltrované dotazy odeslat. Dále je pro spuštění programu zapotřebí dodat soubor s filtrovanými doménami. Tento soubor musí být textového typu a musí být lokálně uložen.

Při běhu program nic nevypisuje, pro výpis paketů a běhu programu je nutné program spustit s parametrem -v.

# Uvedení do problematiky

Pro implementaci programu, který se má chovat jako DNS server, je zapotřebí nejprve nastudovat problematiku tohoto protokolu a způsoby, jakým je možné takovýto program vytvořit.

Osobně jsem začal podrobným nastudováním normy tohoto protokolu[1]. Zde jsem se dozvěděl, jak vypadá DNS paket a z čeho se skládá. Jeho hlavička má velikost 8 bajtů a mimo jiné udává, zda se jedná o dotaz nebo odpověď. Také obsahuje tzv. response code, který udává typ odpovědi.

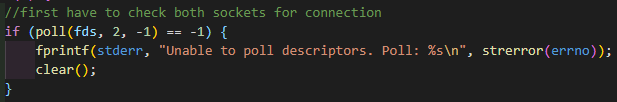
Následně jsem nastudoval, jakým způsobem se implementují síťové aplikace, využívající tzv. sockety[2]. Základem takovéto aplikace je otevřít socket pro příchozí spojení. Následně je vhodné vytvořit nekonečný cyklus pro příjem zpráv a jejich zpracování. To odpovídá chování serveru. V mém projetu jsem využil sockety na bázi UDP připojení.

# Implementace

## Základní údaje

Pro psaní programu jsem zvolil programovací jazyk C. Mezi nejdůležitější použité knihovny patří sys/socket.h pro práci se sokety a arpa/nameser.h pro použití struktury DNS hlavičky. Obě knihovny jsou součástí standardu POSIX a fungují na všech Unix systémech.  
Samotný kód se nachází v jednom souboru dns.c. Celý program je rozložen na několik hlavních částí, které jsou popsány níže.

## Funkce main()

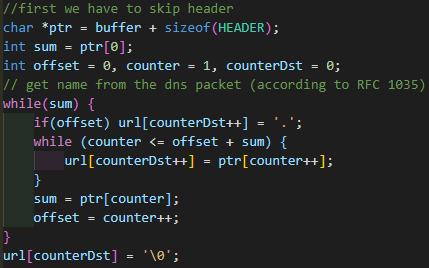
Hlavní část programu, funkce main(), má za úkol zpracování argumentů programu, otevření socketů pro příchozí a odchozí spojení a následné spuštění serveru. S pomocí funkce signal() nastavuje, jak má program správně skončit v případě přerušení systémem nebo uživatelem.  
Pro zpracování argumentů využívám funkci getopt(). Součástí tohoto zpracování argumentů je i funkce printHelp(), která má za úkol vytisknout informace o používání programu.  
O otevření socketů se starají funkce socket() a bind(). Následně, s použitím poll(), spustím nekonečný cyklus reprezentující server. Tuto funkci používám, abych dokázal zachytit všechna spojení.

Obrázek 1 - Použití funkce poll() v nekonečném cyklu

Dál v tomto cyklu se podle typu paketu rozhodne, zda se jedná o dotaz či odpověď, a spustí se samotná kontrola paketu.   
Pokud se jedná o dotaz a program nezjistí žádnou chybu, uloží se adresa a port odesílatele a dotaz se přepošle na skutečný DNS server pomocí funkce sendto(). Když program objeví v dotazu chybu, přetvoří tento paket na chybovou odpověď, kterou odešle tazateli.

V případě, že se jedná o odpověď, se zkontroluje a pošle správnému tazateli zpět.

## Funkce getDnsRequestData()

Tato funkce se stará o získání jména tázaného serveru, typu a třídy z obdrženého DNS paketu. Je zajímavá především algoritmem, jenž projde část paketu, kde se nachází jméno serveru. Podle standardu je jméno rozděleno do částí podle toho, kde se v něm nachází tečky. Na tomto místě je číslo, které udává, kolik znaků se nachází v následující části. Celé jméno je zakončeno tečkou.

Obrázek 2 - algoritmus procházení jména v DNS paketu

## Funkce isBlacklisted()

Tato funkce je velmi jednoduchá a efektivní. Má za úkol zjistit, zda je dotazovaná doména v seznamu filtrovaných domén. Jako parametr přijme jméno domény a následně s pomocí cyklu for() začne procházet celý blacklist, dokud na tuto doménu nenarazí. Důležité je, že funkce zachytí i poddomény těch zakázaných. To je umožněno pomocí knihovní funkce strstr(), která najde výskyt jednoho řetězce v druhém. Takto například odfiltruji dotaz na fit.vutbr.cz i v případě, že mám ve filtrovaných doménách pouze vutbr.cz.

## Funkce getDnsFilter()

Obrázek 3 - Funkce isBlacklisted()

Důležitou součástí programu je seznam filtrovaných domén. Tato funkce načte soubor a domény v něm uloží do proměnné v programu. Je důležité zmínit, že funkce bude načítat do té doby, než mu fukce realloc() nepovolí rozšířit paměť. Také je důležité, že existuje limit na délku jednoho doménového serveru, konkrétně se jedná o 512 bajtů.  
Samotná funkce spravuje dynamické pole v rámci struktury blacklist a načítá do něj postupně všechny řádky (doménová jména) ze zadaného souboru. Samozřejmě ignoruje řádky, které začínají znakem „#“.

## Funkce clear()

Poslední z nejdůležitějších součástí programu je funkce clear(). Jedná se o pomocnou funkci, která se spustí, kdykoliv je program ukončen. Její spuštění i při „násilném ukončení“ (sigterm) zajišťuje již zmíněná funkce signal().  
Tato funkce hlavně uvolňuje alokovanou paměť a zavírá sockety.

# Příklady výstupu programu

# Testování programu

# Závěr

# Zdroje

1. Domain Implementation and Specification [online]. 1987 [cit. 2020-11-14]. Dostupné z: <https://tools.ietf.org/html/rfc1035>
2. Socket Programming in C/C++ [online]. 2019 [cit. 2020-11-14]. Dostupné z: <https://www.geeksforgeeks.org/socket-programming-cc/>