VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

ROZŠÍŘENÍ APLIKACE DPDK DNS PROBE

THE DPDK DNS PROBE APPLICATION EXTENSION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PAVEL DOLEŽAL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ROMAN VRÁNA**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Doležal Pavel, Bc.**
Program: Informační technologie Obor: Počítačové sítě a komunikace
Název: **Rozšíření aplikace DPDK DNS Probe**
The DPDK DNS Probe Application Extension
Kategorie: Počítačové sítě
Zadání:

1. Nastudujte framework DPDK, protokol DNS a jeho oficiální rozšíření.
2. Seznamte se s aplikací DPDK DNS Probe vyvíjené v rámci smluvního výzkumu se společností CZ.NIC.
3. Navrhněte možná vylepšení a optimalizace aplikace.
4. Implementuje navržené změny.
5. Otestujte implementované změny.
6. Ověřte, jaký dopad měly implementované změny na výkon aplikace.
7. Zhodnoťte dosažené výsledky a diskutujte další možnosti rozšíření práce.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Vrána Roman, Ing.**
Konzultant: Dražil Jan, Ing., UPSY FIT VUT
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 26. října 2018

Abstrakt

Předmětem této diplomové práce je rozšíření aplikace DPDK DNS sonda pro monitorování DNS provozu ve vysokorychlostních sítích. V práci je popsán framework DPDK, který slouží k rychlému zpracování paketů. Je popsána architektura systému DNS a fungování jeho jednotlivých komponent. Dále jsou popsány základní principy transportního protokolu TCP. Představen je návrh a implementace efektivního parsování paketů DNS pro optimalizaci aplikace DPDK DNS sonda. Dále je představen návrh a implementace extrakce DNS zpráv posílaných přes protokol TCP pro export statistik provozu. Výkon aplikace byl otestován pomocí generátoru síťového provozu Spirent.

Abstract

This master's thesis is focused on extension of the DPDK DNS Probe application that monitors DNS traffic in high speed networks. It presents framework DPDK, which can be used for fast packet processing. General architecture of the DNS system is described as well as details of its components. Basic principles of transport protocol TCP are described. It introduces an effective design and implementation of DNS packet parsing to optimize DPDK DNS Probe's performance. It also introduces a design and implementation of processing DNS messages sent over TCP for export of traffic statistics. The application's performance was tested using a high speed traffic generator Spirent.

Klíčová slova

služba DNS, framework DPDK, TCP, vysokorychlostní sítě, CZ.NIC, monitorování DNS, knihovna libknot, parsování paketů DNS, DNS přes TCP

Keywords

DNS, DPDK framework, TCP, high speed networks, CZ.NIC, DNS monitoring, libknot library, DNS packet parsing, DNS over TCP

Citace

DOLEŽAL, Pavel. *Rozšíření aplikace DPDK DNS Probe*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Roman Vrána

Rozšíření aplikace DPDK DNS Probe

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Romana Vrány. Jako konzultant mi velmi pomohl pan Ing. Jan Dražil. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Doležal
21. května 2019

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Romanu Vránovi a také konzultantovi Ing. Janu Dražilovi za poskytnuté rady, ochotu a připomínky při tvorbě práce.

Obsah

Kapitola 1

Úvod

Bez internetové komunikace si dnes již život ani neumíme představit. Pro člověka jednoduché úkony jako zobrazení webové stránky nebo stažení videa ze vzdáleného úložiště je ale nutné vhodným způsobem vysvětlit našim zařízením. Ty neznají názvy jako *seznam.cz* nebo *youtube.com*. Síťová zařízení komunikují v řeči čísel a je tedy nutné nějakým způsobem převést názvy srozumitelné lidem na názvy srozumitelné strojům.

K tomuto účelu se využívá služby Domain Name System (zkráceně DNS), která převádí textové názvy na číselné a naopak. Služba DNS je celosvětovou databází IP adres a jim odpovídajících textových názvů, bez které by bylo použití internetových služeb pro člověka mnohem komplikovanější. Jelikož voláním služby DNS začíná velká část internetové komunikace, je nutné zaručit její nepřerušenou dostupnost. Pro tyto účely je vhodné službu DNS pečlivě monitorovat a vést si podrobné statistiky o jejím provozu.

V kapitole 2 práce představuje principy fungování služby DNS a její architekturu. Rozebrán je protokol DNS a formát jeho zpráv. Kapitola 3 popisuje základní principy transportního protokolu TCP, zejména ty vztahující se k přenosu DNS zpráv. Kapitola 4 popisuje framework DPDK, který poskytuje rozhraní pro implementaci aplikací zpracovávajících pakety ve vysokorychlostních sítích. Kapitola 5 představuje aplikaci DPDK DNS sonda, která se zabývá monitorováním DNS provozu ve vysokorychlostních sítích a exportem podrobných statistik tohoto provozu. Kapitola 6 se zabývá měřením rychlosti parsování DNS paketů pomocí knihovny *libknot* v aplikaci DPDK DNS sonda a prvotními pokusy o optimalizaci. Dále je zde představen návrh efektivní implementace parsování DNS paketů pro spolehlivý export statistik v aplikaci DPDK DNS sonda. V kapitolách 7 a 8 je popsána implementace a následné testování optimalizovaného parsování DNS paketů bez využití knihovny *libknot*. Kapitola 9 představuje návrh rozšíření aplikace DPDK DNS sonda o podporu zpracování DNS provozu přes transportní protokol TCP. V kapitolách 10 a 11 je poté popsána implementace a testování tohoto rozšíření. V kapitole 12 jsou nastíněny možnosti dalšího vývoje aplikace DPDK DNS sonda.

Kapitola 2

DNS

Překlad doménových jmen patří mezi nezbytné součásti internetu, bez kterých se neobejde žádná komunikace mezi uživateli. Nefunkčnost překladu doménových jmen zjistíme takřka okamžitě, když nedokážeme načíst žádnou webovou stránku nebo poslat email.

V této kapitole je vysvětlena architektura DNS (Domain Name System). Seznámíme se zde se základními stavebními prvky systému a způsobem jejich komunikace. Dále se seznámíme se způsobem uložení dat a přístupem k nim. Rozebereme také jednotlivé typy DNS záznamů, které systém spravuje. Informace v této kapitole vycházejí z knihy *Síťové aplikace a jejich architektura* [22].

2.1 Služba DNS

Základním úkolem služby DNS je mapování doménových jmen (např. fit.vutbr.cz) na IP adresy (147.229.9.23). IP adresa je jednoznačný identifikátor, který má přiřazeno každé síťové rozhraní na internetu. Z pohledu uživatele je ale používání IP adres pro přístup k připojeným stanicím nevhodné. Pro síťová zařízení je na druhou stranu nevhodné identifikovat se pomocí textových řetězců. Číselná hodnota se velmi hodí pro porovnávání a rychlé prefixové vyhledávání.

Jmenné ekvivalenty IP adres se začaly objevovat už při zavedení IP adres v 70. letech 20. století. Systém DNS se stal standardem pro celosvětovou databázi IP adres a jim odpovídajících textových řetězců – doménových jmen.

Služba DNS zahrnuje databázi všech doménových jmen a příslušných IP adres. Dále definuje způsob, jak k těmto datům lze přistupovat. Kvůli objemu této databáze byl použit distribuovaný model, kdy je databáze rozprostřena mezi více koncových stanic. Těmto stanicím se říká doménové servery (jmenné servery) nebo také *nameservery*. K zjištění IP adresy z doménového jména musíme poslat dotaz na doménový server a vyčkat na odpověď. Tomuto procesu se říká rezoluce nebo rozlišení jména (*name resolution*).

Většina síťových aplikací dnes používá systém DNS pro překlad doménových jmen na IP adresy. Tento krok je vždy první činností aplikace, když si vyžádáme připojení na vzdálenou službu a jako cíl zadáme doménové jméno. Aplikace nejprve zašle na doménový server dotaz k překladu na IP adresu. Až poté teprve dojde k pokusu o navázání spojení s cílovou službou pomocí získané IP adresy.

Systém DNS neposkytuje pouze překlad doménových jmen na IP adresy. Mezi základní služby systému DNS patří:

- překlad doménových jmen na IP adresy (pomocí záznamů typu A a AAAA),
- překlad IP adres zpět na doménová jména (pomocí záznamů typu PTR),
- překlad aliasů počítačů, překlad na tzv. kanonická jména (pomocí záznamů typu CNAME),
- identifikace poštovního serveru pro danou doménu (pomocí záznamů typu MX),
- podpora rozložení zátěže mezi více doménových serverů,
- sdílení informací v rámci globálního prostoru jmen či delegování správy na jednotlivé subjekty (pomocí záznamů typu NS).

Původně se pro mapování doménových jmen na IP adresy používal jen jeden soubor HOSTS.TXT. Tento soubor spravovala organizace NIC (Network Information Center). Data z tohoto souboru byla šířena sítí mezi ostatní koncové stanice v pravidelných aktualizacích pomocí služby FTP [29]. S rozšířením lokálních sítí se záznamy musely udržovat na lokální úrovni. Aby byly změny v lokální síti viditelné v Internetu, bylo nutné lokální mapování předávat do NIC. Z tohoto systému se postupně vyvinul koncept distribuované databáze s doménovými servery.

Jako primární cíl DNS bylo stanoveno vytvoření konzistentního prostoru doménových jmen odkazujícího na síťová zařízení. Systém DNS byl navržen tak, aby k němu měl přístup nejen IP protokol, ale i jiné rodiny protokolů a aplikace. Hlavním úkolem je tedy zajištění snadného přístupu k uloženým informacím ze všech zařízení v síti. Hlavní požadavky na databázi DNS se dají shrnout v následujících bodech:

- Velikost celkové databáze je závislá na počtu zařízení v síti, který v budoucnu poroste.
- Data v databázi se většinou nemění moc často. Systém však musí být schopný zareagovat na drobné změny velmi rychle (v řádech sekund a minut).
- Přístup k datům má prioritu oproti jejich aktuálnosti. I když není možné data aktualizovat, musíme zajistit alespoň běh služby.
- Originální data jsou udržována lokálně v tzv. primárních záznamech (master files). Tyto záznamy se šíří na další zařízení pomocí systému DNS. Formát záznamů je textový a spravuje je lokální administrátor.

V systému DNS jsou zahrnuta všechna zaregistrovaná doménová jména a jejich mapování na příslušné IP adresy. Není však třeba vytvářet doménová jména pro všechny IP adresy. Například pokud je IP adresa využita pouze pro systémové služby, není třeba jí přiřazovat doménové jméno. Některé IP adresy naopak mohou mít přiřazeny více doménových jmen.

2.2 Architektura DNS

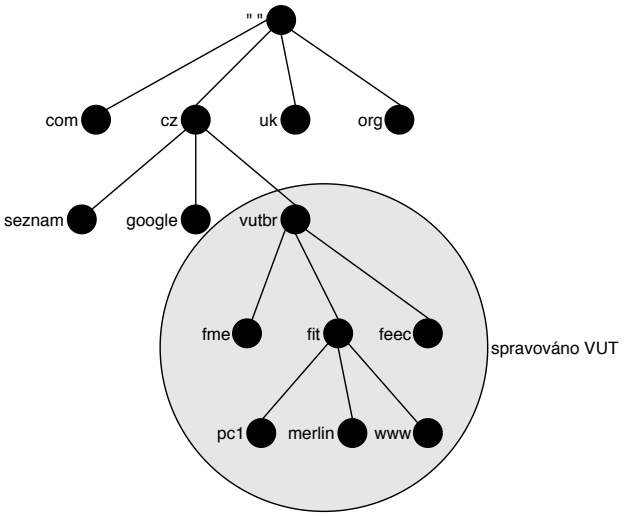
Systém DNS tvoří tři hlavní komponenty – prostor doménových jmen, doménové servery a resolver. Do prostoru doménových jmen zahrnujeme strukturu, uspořádání a přístup k datům v systému DNS. Doménové servery tato data ukládají ve svých lokálních databázích.

Resolver slouží klientským stanicím k přístupu k datům v systému DNS. Nyní si jednotlivé komponenty podrobněji popíšeme.

2.2.1 Prostor doménových jmen

Prostor doménových jmen je definován jako hierarchicky uspořádaný kořenový strom doménových jmen. Strom DNS má jeden kořen. Jednotlivé uzly stromu jsou identifikovány textovým řetězcem maximální délky 63 znaků. Kořen stromu má speciální textový identifikátor nulové délky. Uzly se stejným bezprostředním předchůdcem musejí být pojmenovány různě, aby je bylo možné rozlišit.

Doménové jméno je cesta mezi kořenem stromu DNS a uzlem, který tvoří vrchol domény. Jednotlivé uzly jsou v této cestě odděleny tečkou. Mezi speciální domény stromu DNS patří doména první úrovně (TLD, Top Level Domain), což je doména se vzdáleností jedna od kořene stromu. Listy stromu poté označují konkrétní síťová zařízení v dané doméně, například *merlin* nebo *eva* v doméně *fit.vutbr.cz.*. Jednotlivé části podstromu doménových jmen jsou uloženy a spravovány na lokálních doménových serverech, které dohromady tvoří systém DNS. Doménu *.cz* například spravuje registrátor CZ.NIC.



Obrázek 2.1: Strom doménových jmen v DNS. Převzato z [22].

2.2.2 Doménové servery

Doménový server je zařízení uchovávající data z prostoru doménových jmen. Tento prostor je rozdělen do zón, které jsou rozmístěny na jednotlivé doménové servery. Úkolem doménového serveru je odpovídat na dotazy resolveru. Data jsou na doménovém serveru uchovávána ve formě DNS záznamů (resource records). Záznamy jsou uloženy buď v lokálním souboru nebo jsou načteny z jiného doménového serveru pomocí přenosu zón. Data, která poskytuje a spravuje daný doménový server, se nazývají autoritativní. O data pro jednu zónu se může starat více doménových serverů. V tomto případě musíme rozlišit jednotlivé typy doménových serverů:

- **Primární server**

  Primární server obsahuje úplné záznamy o všech doménách, které spravuje. Tyto záznamy má uloženy v lokálním souboru a jeho odpovědi jsou vždy autoritativní pro spravované domény. Každá doména má právě jeden primární doménový server.

- **Sekundární server**

  Sekundární server získává data od primárního serveru, ale nemusí u sebe uchovávat všechny domény, které primární server spravuje. Soubor se záznamy pro konkrétní doménu se nazývá zónový soubor. Přenos tohoto souboru z primárního na sekundární server probíhá přenosem zón. Sekundární server musí být schopen zajistit pravidelný přenos dat z primárního serveru a aktuálnost dat. Sekundární server je pro danou doménu stejně jako primární server autoritativní.

- **Záložní server** (caching-only server)

  Záložní server funguje jako proxy server mezi resolvery a autoritativními servery. Přijímá dotazy od resolverů a přeposílá je autoritativním serverům. Když dostane nazpět odpověď od některého autoritativního serveru, přepošle ji resolveru a zároveň si ji lokálně uloží pro případné použití v budoucnosti. Tímto způsobem zmírňuje zátěž autoritativním serverům. Odpovědi od záložního serveru ale nejsou autoritativní, protože mohou být neaktuální.

2.2.3 Resolver

Resolver je program na straně klienta, který posílá dotazy na data uložená na doménových serverech. Všechny uživatelské programy, které potřebují data ze systému DNS, používají pro přístup k těmto datům jako prostředníka resolver.
Úkolem resolveru je:

- posílat dotazy na doménové servery,
- zpracovávat odpovědi od doménových serverů,
- předat zpracovaná data danému uživatelskému programu.

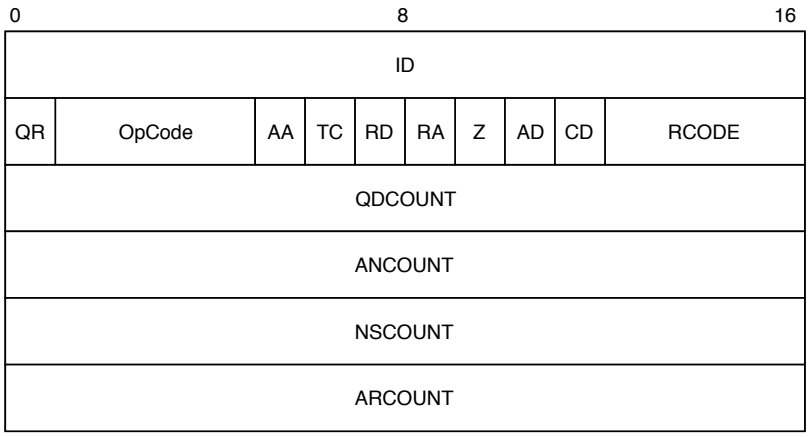
Resolver musí být vždy schopen posílat dotazy na alespoň jeden doménový server. Server mu buď pošle žádaná data nebo jako odpověď vrátí odkaz na další doménový server, který by mohl mít žádaná data k dispozici. Takto může resolver posílat dotazy na další doménové servery. Obvykle je resolver implementován jako rutina operačního systému, ke které přímo přistupují uživatelské programy.

2.3 Protokol DNS

Protokol DNS, popisující veškerou komunikaci v systému DNS, je definován v standardu RFC 1035 [26]. Pro komunikaci ve formátu dotaz-odpověď se většinou používá transportní protokol UDP se serverovým číslem portu 53. Standard povoluje i použití transportního protokolu TCP se stejným číslem portu. Pro DNS pakety využívající transportní protokol UDP je standardem omezena velikost paketu na 512 bytů. Pokud chceme protokolem UDP odeslat větší zprávu, musíme ji rozdělit do několika paketů, ve kterých nastavíme bit TC (TrunCation) v DNS hlavičce paketu. Jelikož protokol UDP poskytuje nespolehlivou komunikaci, aplikace musí v případě ztráty paketu odeslat dotaz na server znovu. Může se

také stát, že UDP pakety dojdou na server v pozměněném pořadí, než je aplikace odeslala. Proto DNS hlavička obsahuje 16-bitové číslo, kterým je každý dotaz jednoznačně identifikován. Pro přenos zón mezi doménovými servery se kvůli spolehlivosti používá transportní protokol TCP.

DNS paket je rozdělen na hlavičku a data. Datovou část DNS paketu tvoří čtyři hlavní části. Jsou to část obsahující dotaz na server (question), část obsahující odpověď serveru (answer), část obsahující informace o autoritativních serverech (authority) a část s doplňujícími informacemi (additional). DNS hlavička se skládá z polí pevné délky, které obsahují informace o počtu záznamů v každé sekci datové části paketu a polí, které obsahují identifikační a doplňující informace o dané transakci aplikace-server. Mezi tyto pole patří již zmiňovaný 16-bitový identifikátor transakce (ID), bit identifikující paket jako dotaz či odpověď (QR), operační kód specifikující typ dotazu (OpCode), příznak určující jestli odpověď přichází od autoritativního serveru (AA), příznak rozdělení zprávy do více paketů (TC), příznaky pro rekurzivní vyhledávání (RD, RA), bity rezervované pro budoucí použití (Z) a kód určující typ odpovědí (RCODE). V RFC 2535 [16] a RFC 3655 [31] byly přidány příznaky AD a CD, které řeší autentizaci obsahu dotazů a odpovědí. Formát DNS hlavičky je zobrazen na obrázku 2.2.



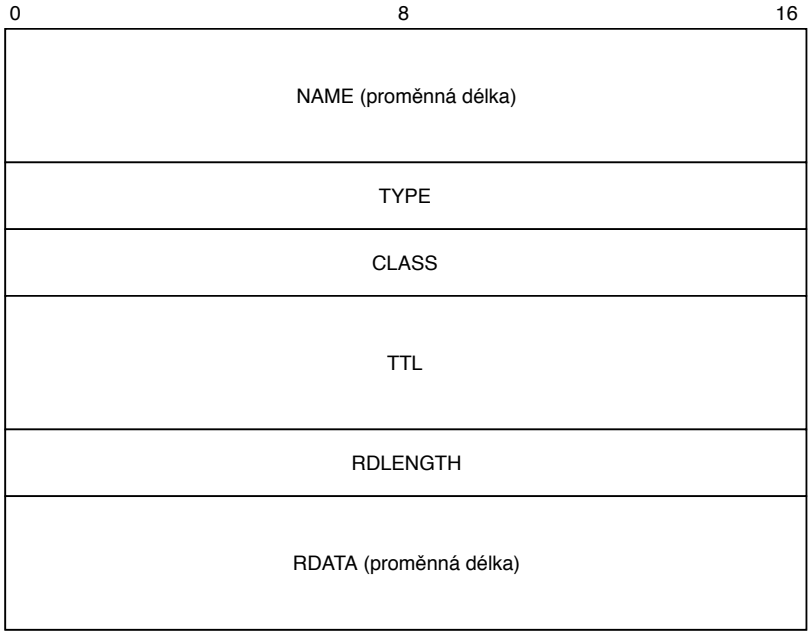
Obrázek 2.2: DNS hlavička paketu

2.3.1 Záznamy DNS

V datové části DNS paketů jsou v jednotlivých sekcích přenášeny tzv. záznamy DNS (Resource Records). Ty slouží v systému DNS k uchovávání informací o prostoru doménových jmen. Jsou uloženy v textové podobě v zónových souborech na doménových serverech. Nejběžnější jsou záznamy typu A, které mapují doménové jméno na odpovídající IP adresu verze 4. Opačné mapování poskytují záznamy PTR. Pro mapování doménových jmen na IP adresy verze 6 se používají záznamy typu AAAA. Všechny typy záznamů DNS jsou definovány v několika různých standardech, např. RFC 1034 [25], RFC 1035 [26], RFC 1183 [17], RFC 3401 [23], RFC 3597 [19], aj. Typy záznamů se liší od běžně používaných po spíše experimentální, které se v praxi nikdy nerozšířily (Responsible Person, Mailbox).

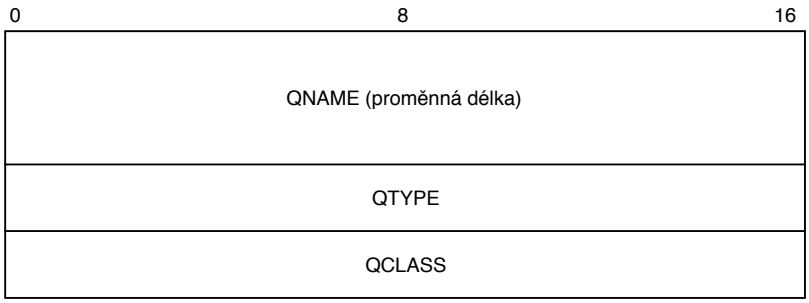
Všechny typy záznamů mají obecně stejný formát definovaný v standardu RFC 1035, který je zobrazen na obrázku 2.3. Každý záznam DNS povinně obsahuje všechny zobrazené položky. Položka NAME obsahuje název uzlu ve stromu DNS, ke kterému se daný záznam vztahuje. Délka této položky je tedy proměnlivá. Následuje 16-bitová položka TYPE, která

určuje typ daného záznamu DNS. Další 16-bitovou položkou záznamu je CLASS, která definuje třídu dat uložených v položce RDATA. Položka TTL (Time-To-Live) je 32-bitová a určuje dobu platnosti daného záznamu. Následuje 16-bitová hodnota RDLENGTH, která určuje velikost následující položky RDATA v bytech. Položka RDATA obsahuje hodnotu záznamu, který nás zajímá. Různé typy záznamů DNS se liší v obsahu položky RDATA. Liší se jak počtem polí této položky, tak jejich délkou a významem.



Obrázek 2.3: Formát záznamu DNS

Speciální sekcí datové části paketu DNS je první část obsahující dotaz na doménový server. Tato sekce jako jediná nedodržuje výše popsaný formát záznamů DNS, ale má svůj vlastní zjednodušený formát zobrazený na obrázku 2.4. Tento formát obsahuje pouze položky QNAME, QTYPE a QCLASS, které mají stejný význam jako jejich odpovídající alternativy v záznamu DNS.

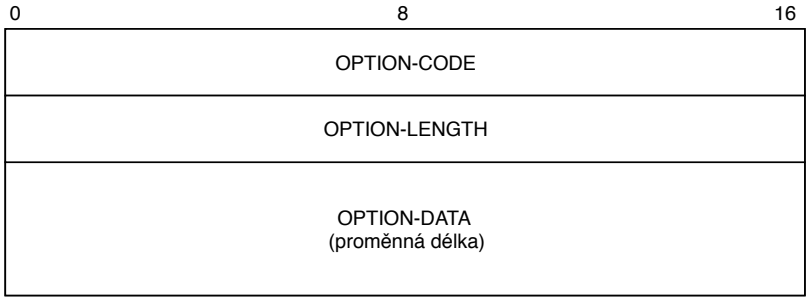


Obrázek 2.4: Formát DNS dotazu na doménový server

2.3.2 EDNS záznam

EDNS je speciálním typem DNS záznamu, který se nachází v datové části DNS paketu v sekci s doplňujícími informacemi. Jeho formát je popsán v RFC 6891 [7] a poté dále rozvíjen v RFC 6975 [4], RFC 7871 [3], aj. Jelikož je DNS dnes masivně rozšířeným protokolem, na kterém závisí správné fungování internetu, vznikla snaha přenášet pomocí něj i další informace, na které nebyl původně navržen. K tomuto účelu byl vytvořen záznam EDNS.

Formát EDNS záznamu se neliší od formátu ostatních záznamů zobrazeném na obrázku 2.3. Samotná dodatečná data jsou přenášena v poli RDATA, které je rozděleno na jednotlivé option položky, jejichž formát je zobrazen na obrázku 2.5. Pole OPTION-CODE udává typ přenášených data v dané položce a pole OPTION-LENGTH poté jejich délku v bytech. Pole OPTION-DATA obsahuje samotná data, jejichž formát je v závislosti na typu položky definován ve standardu. Mezi přenášené informace může patřit např. seznam klientem podporovaných hashovacích algoritmů pro bezpečný přenos DNS záznamů technologií DNSSEC, nebo síť, ze které původně vzešel dotaz.



Obrázek 2.5: Formát EDNS option položky

Kapitola 3

TCP

Transportní protokol UDP je pro přenos DNS zpráv používán z důvodu jednoduchosti a malé režie. Nezaručuje ovšem spolehlivý přenos a tak může dojít ke ztrátám zpráv při přenosu po síti. Dalším omezením přenosu DNS přes UDP je omezení maximální velikosti paketu na 512 bytů. Oběma těmto nedostatkům je možné se vyhnout pomocí transportního protokolu TCP, jehož použití pro přenos DNS zpráv je definováno v RFC 1035 [26] a dále upřesněno v RFC 7766 [8].

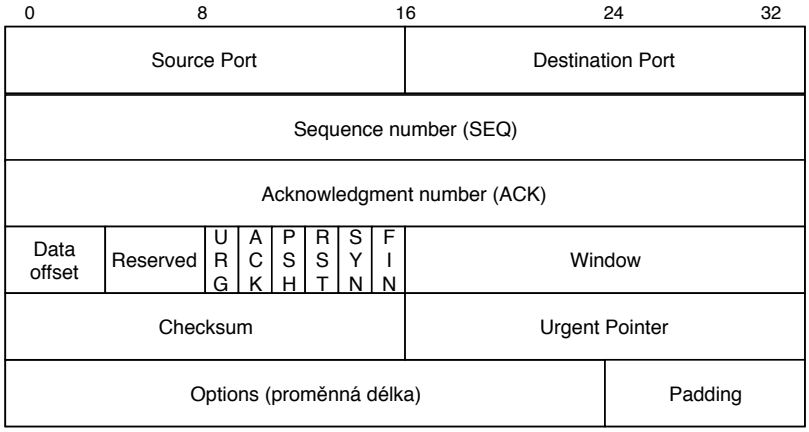
V této kapitole jsou vysvětleny obecné principy transportního protokolu TCP, které se vztahují k přenosu DNS zpráv. Seznámíme se se způsobem zajištění spolehlivého přenosu dat, metodou ustanovení a ukončení spojení mezi klientem a serverem aj. Informace v této kapitole vycházejí z knihy *TCP/IP architecture, design and implementation in Linux* [30] a příslušných standardů RFC, zejména RFC 793 [28].

3.1 TCP hlavička

TCP segment se skládá z hlavičky a datové části. Hlavička transportního protokolu TCP bezprostředně obaluje zprávu DNS v rámci paketu, která tedy tvoří datovou část TCP segmentu. TCP hlavička obsahuje informace definující dané spojení, informace pro kontrolu stavu daného spojení a pole pro validaci TCP hlavičky. Základní velikost TCP hlavičky je 20 bytů. Může ale být rozšířena o další volitelné položky (options). Velikost TCP hlavičky musí být vždy zarovnána na násobek 4 bytů. Obecný formát TCP hlavičky je zobrazen na obrázku 3.1.

- **Čísla portů.** TCP spojení je jednoznačně určeno čtveřicí hodnot – zdrojová IP adresa, cílová IP adresa, zdrojový port a cílový port. Zdrojový a cílový port jsou první dvě pole v TCP hlavičce. Každé má velikost 2 byty. Čísla portů jednoznačně určují přístupový bod daného TCP spojení na každém konci spojení.

- **Sekvenční číslo (SEQ).** Sekvenční číslo o velikosti 4 byty určuje offset prvního bytu datové části segmentu od začátku datového toku dané strany TCP spojení. Nereflektuje ovšem skutečný počet odeslaných bytů v rámci dané strany spojení. Sekvenční číslo je offset od počátečního sekvenčního čísla dané strany spojení (ISN). Počáteční sekvenční číslo, ISN, je pseudonáhodně generováno při ustanovení spojení na obou jeho koncích. Pro dané spojení jsou počáteční sekvenční čísla na obou koncích spojení unikátní. Primární důvod pro unikátnost ISN je potřeba rozlišit případné zpožděné pakety předcházejícího spojení na stejných portech od spojení aktuálního. Při přete-



Obrázek 3.1: TCP hlavička paketu

čení maximálního rozsahu sekvenčních čísel během spojení dojde k počítání dalších sekvenčních čísel opět o nuly. Sekvenční čísla pomáhají udržet integritu posílaných dat a identifikovat data, která už byla v minulosti zaslána.

- **Potvrzovací číslo (ACK).** Potvrzovací číslo o velikosti 4 byty je protějškem sekvenčního čísla. Jelikož je TCP protokol zaručující spolehlivý přenos dat, musí si každá strana spojení udržovat přehled o datech, která byla doručena druhé straně. K tomuto účelu slouží potvrzovací číslo. Přijímající strana potvrzuje pomocí potvrzovacího čísla poslední přijatý byte v rámci datového toku dané strany spojení. Hodnota potvrzovacího čísla se stejně jako u sekvenčního čísla odvíjí od hodnoty počátečního sekvenčního čísla. Skutečná posílaná hodnota potvrzovacího čísla je vždy ve skutečnosti o jedničku větší než číslo odpovídající poslednímu přijatému bytu v rámci datového toku. Odpovídá tedy následujícímu sekvenčnímu číslu, které je daná strana spojení připravena přijmout. Pokud přijímací straně spojení dojdou data se sekvenčními čísly, která ještě neočekává, nebudou tato data odesílající straně potvrzena pomocí paketu s příslušným potvrzovacím číslem. Tímto způsobem odesílající strana nakonec zjistí, že některé odeslané segmenty nebyly přijaty a zajistí jejich opětovné zaslání.
- **Data offset.** 4-bitové pole, které indikuje celkovou velikost TCP hlavičky ve slovech (4 byty). Normální velikost TCP hlavičky je 20 bytů, ale kvůli možnosti přidání položek s dodatečnými informacemi může její velikost narůst až na 60 bytů.
- **Rezervované bity.** Pole 6 bitů které jsou rezervované pro budoucí použití.
- **TCP příznaky.** 6-bitové pole příznaků vztahujících se k řízení stavu daného TCP spojení.
  - *URG:* Tento příznak indikuje, že je v hlavičce nastaven ukazatel naléhavých dat, která by měla být zpracována co nejrychleji.
  - *ACK:* Tento příznak indikuje, že v hlavičce bylo odesílatelem nastaveno potvrzovací číslo. Kromě prvního SYN segmentu spojení je tento příznak vždy nastaven, protože nijak neškodí, když přijatá data potvrdíme druhé straně víckrát.
  - *PSH:* Příznak indikující, že data v tomto segmentu by měla být zpracována prioritně.

  - *RST:* Tímto příznakem dává odesílající strana najevo, že chce okamžitě ukončit spojení bez provedení formální ukončující procedury.
  - *SYN:* Tento příznak je nastaven během fáze navazování spojení, kdy si obě strany spojení posílají svoje počáteční sekvenční čísla.
  - *FIN:* Tímto příznakem dává odesílající strana najevo, že chce formálně ukončit dané spojení.

- **Velikost okna.** 2-bytové pole indikující velikost bufferu na straně příjemce pro přijímaná data. Tento buffer se na straně příjemce plní datovou částí TCP segmentů a je vyprazdňován příslušnou aplikací, které data náleží. Když jsou data posílána rychleji, než je aplikace u příjemce stačí zpracovat, buffer se postupně zaplní a velikost okna klesne až na nulu. V tomto momentě odesílatel přestane posílat další data dokud příjemce nepošle zprávu, že bylo uvolněno místo v bufferu. Každá strana spojení deklaruje velikost svého okna při navazování spojení a poté ji průběžně aktualizuje během průběhu spojení.

- **Kontrolní součet.** Toto 2-bytové pole příjemci indikuje zachování integrity přijatého segmentu během cesty sítí. Kontrola integrity zahrnuje jak TCP hlavičku, tak datovou část segmentu. Toho je dosaženo tak, že jako kontrolní součet vypočteme sumu celého segmentu po 2-bytových blocích a vezmeme jednotkový doplněk tohoto součtu. Příjemce tento výpočet provede také a porovná ho s kontrolním součtem v hlavičce.

- **Ukazatel naléhavých dat.** Toto 2-bytové pole je vyplněno, pokud je v hlavičce nastaven příznak *URG*. Určuje offset od sekvenčního čísla daného segmentu, kde se nachází data k prioritnímu zpracování.

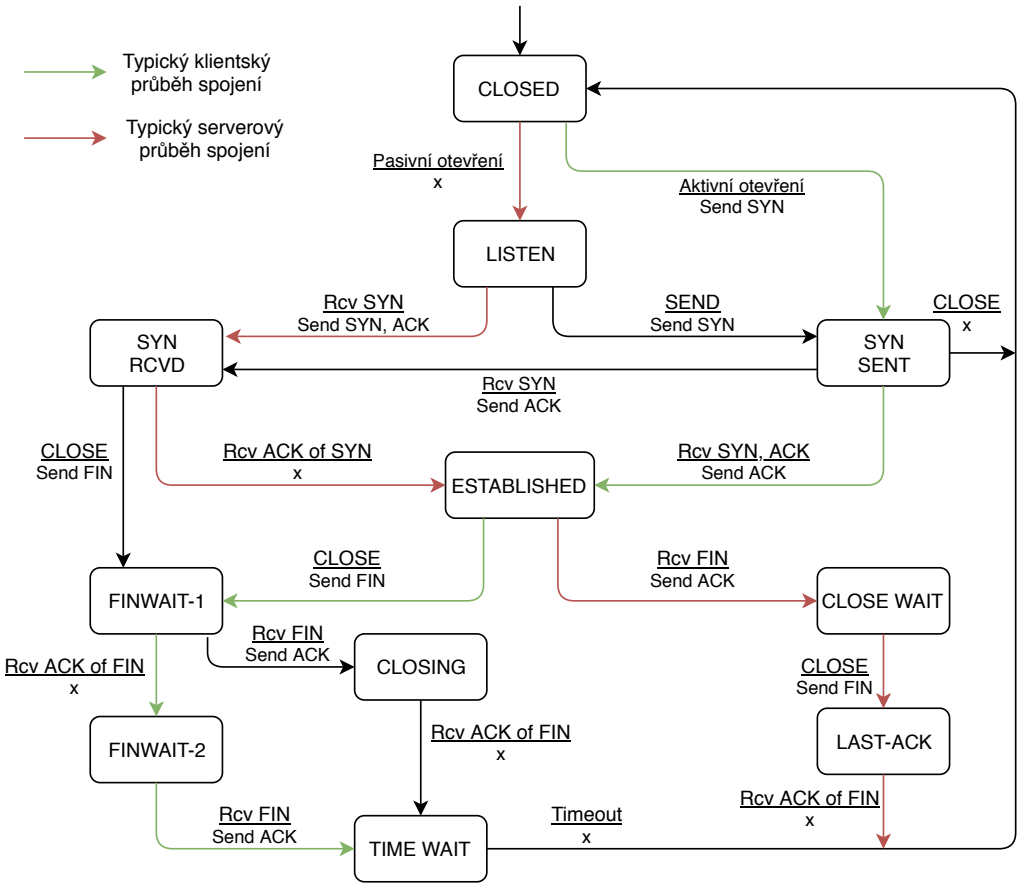
- **Volitelné položky.** Hlavička může být volitelně rozšířena o doplňující informace. Mezi tyto informace patří např. maximální velikost přenášeného segmentu, upřesnění velikosti okna, časové razítko nebo selektivní potvrzování přijatých segmentů.

- **Zarovnání.** TCP hlavička musí být vždy zarovnána na násobek čtyř bytů. Při použití volitelných položek se ale může stát, že velikost hlavičky se bude od tohoto násobku lišit. K zarovnání na příslušný násobek čtyř bytů je zbytek hlavičky doplněn nulami.

3.2 Průběh TCP spojení

K zajištění spolehlivosti TCP spojení je jeho celý průběh pečlivě popsán stavovým automatem. Před samotným zasíláním dat musí být spojení nejdříve formálně navázáno a taktéž po ukončení datového přenosu musí být spojení korektně uzavřeno. Stavový automat pro klientskou i serverovou stranu spojení je zobrazen na obrázku 3.2.

Samotný přenos dat probíhá téměř výhradně ve stavu ESTABLISHED, který nastane po formálním navázání spojení a trvá až do začátku formálního ukončení spojení. Na straně klienta dochází zpravidla k tzv. aktivnímu otevření spojení, kdy klient spojení iniciuje odesláním prvního paketu spojení s nastaveným příznakem SYN. Tímto přechází do stavu SYN-SENT, který indikuje začátek spojení na straně klienta. U serveru dochází zpravidla k tzv. pasivnímu otevření spojení. Server je původně ve stavu LISTEN a naslouchá na daném portu. Při příchodu paketu s nastaveným příznakem SYN se přepne do stavu SYN-RCVD, čímž indikuje začátek formální procedury navázání spojení.

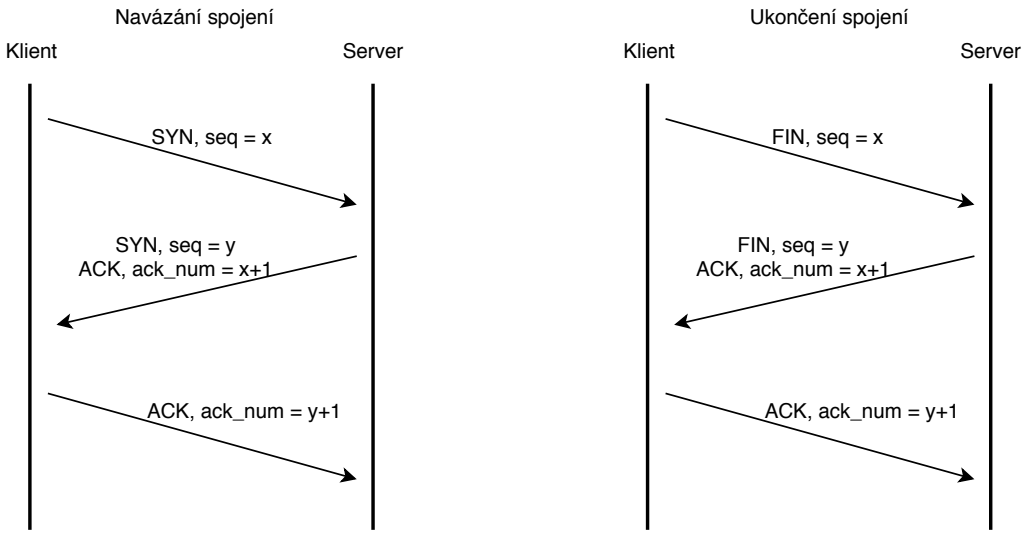


Obrázek 3.2: Konečný automat TCP spojení

Typický průběh navázání a ukončení spojení je zobrazen na obrázku 3.3. Navázání spojení probíhá tak, že strana, která spojení iniciuje (typicky klient), zašle paket s prázdným segmentem, nastaveným příznakem SYN a v TCP hlavičce do pole pro sekvenční číslo nastaví počáteční sekvenční číslo ISN. Druhá strana (typicky server) odpoví také paketem s prázdným segmentem, ve kterém nastaví příznak ACK. Potvrzovací číslo v TCP hlavičce je nastaveno na hodnotu ISN + 1, i když je datová část segmentu prázdná. Tímto se druhé straně indikuje zaregistrování příznaku SYN v předchozím paketu. Druhá strana taktéž nastaví příznak SYN a do pole pro sekvenční číslo nastaví svoje počáteční sekvenční číslo ISN. Navázání spojení je poté dokončeno iniciátorem, který po obdržení paketu s příznakem SYN od druhé strany zašle zpět paket s nastaveným příznakem ACK a jako potvrzovací číslo nastaví hodnotu ISN + 1 pro ISN druhé strany spojení. Tomuto procesu navázání spojení se říká three-way handshake.

Ukončení spojení probíhá na podobném principu jako navázání spojení. Strana, která chce ukončit spojení, pošle prázdný segment s nastaveným příznakem FIN. Druhá strana na tento paket odpoví paketem s nastaveným příznakem ACK a potvrzovacím číslem v TCP hlavičce nastaveným na hodnotu o jedna větší, než bylo sekvenční číslo v paketu s příznakem FIN. Tímto se straně iniciující ukončení dává najevo, že příznak FIN byl druhou stranou zaznamenán. Druhá strana může nastavit svůj příznak FIN již v paketu potvrzujícím zaregistrování příznaku FIN od iniciující strany. Pokud ale ještě má data k odeslání, může nejdříve odeslat tato data, která nejsou nijak omezena, co se týče jejich množství. Až

když už nemá druhá strana žádná data k odeslání, pošle také prázdný segment s nastaveným příznakem FIN. Strana iniciující ukončení na tento paket odpoví prázdným paketem s nastaveným příznakem ACK a potvrzovacím číslem nastaveným na hodnotu o jedna větší, než bylo sekvenční číslo v paketu s příznakem FIN. Po této proceduře považují obě strany spojení za uzavřené.



Obrázek 3.3: Schéma navázání a ukončení TCP spojení

3.3 Řízení datového toku

Protokol TCP garantuje spolehlivý přenos dat na nespolehlivé síti. Když odešleme paket po síti, může dojít k jeho ztrátě během přenosu, paket může dojít v jiném pořadí, než byl odeslán nebo příjemce může být zahlcen a neschopen paket přijmout. Aplikace posílající data po síti se nestarají o jejich spolehlivé doručení. Pouze zapíší data na příslušný soket a TCP se o jejich spolehlivé doručení postará za ně pomocí mechanismů spolehlivého řízení toku.

Spolehlivé řízení toku v TCP spočívá v tom, že odesílatel si dává pozor, aby neposílal data po síti rychleji, než je příjemce stačí konzumovat. TCP ještě implementuje další podobný mechanismus, a to prevenci zahlcení. Rozdíl oproti řízení toku spočívá v tom, že prevence zahlcení usiluje o to, aby odesílatel nezahltil samotnou síť, po které jsou data odesílána.

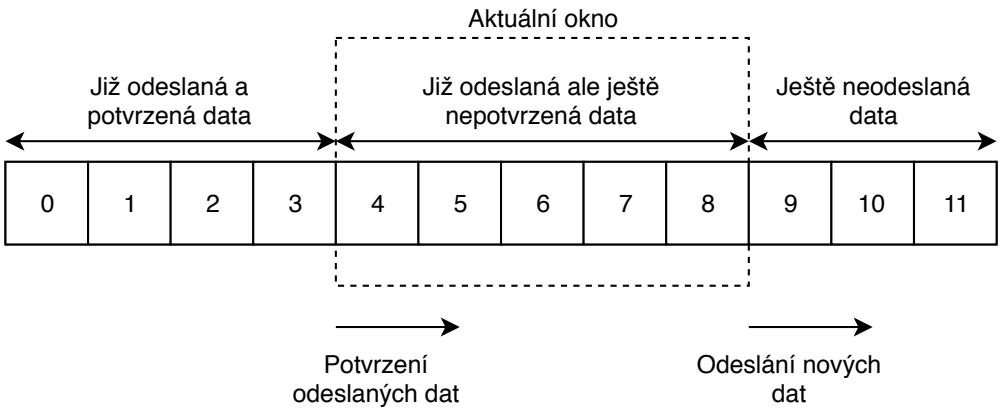
Odesílající aplikace zapíše data na soket, ze kterého jsou uloženy do odesílacího bufferu. TCP bere postupně data z odesílacího bufferu a zabaluje je do segmentů, které posílá po síti. Na straně příjemce jsou data ze segmentů ukládána do přijímacího bufferu. Díky sekvenčním číslům v každém segmentu je příjemce schopen sestavit data do bufferu ve správném pořadí a dokáže identifikovat, když některá část dat chybí. TCP na straně příjemce poté aplikacím vždy zpřístupňuje pouze již celistvý proud dat. Pokud je v přijímacím bufferu mezera, kde některá část dat chybí, data přijatá za touto mezerou nesmějí být zpřístupněna aplikaci, dokud nepřijdou segmenty s chybějícími daty.

Přijímací buffer na straně příjemce se tedy neustále dynamicky naplňuje a vyprazdňuje. Spolehlivé řízení toku zajišťuje, že odesílatel nebude posílat další data, když se buffer na

straně příjemce zahltí. K tomuto účelu TCP používá položku Velikost okna v TCP hlavičce paketu popsané v sekci 3.1. Tato položka udává dostupné volné místo v přijímacím bufferu na straně příjemce. Po každém přijetí nových dat musí příjemce poslat paket s nastaveným příznakem ACK a vyplněným potvrzovacím číslem, které indikuje korektní přijetí nových dat. V tomto paketu také příjemce v položce Velikost okna vždy nastaví aktuální dostupné místo v přijímacím bufferu.

TCP používá protokol klouzavého okna k určení, kolik dat může mít aktuálně uprostřed přenosu. To jsou data, která byla odeslána po síti, ale jejich doručení ještě nebylo potvrzeno druhou stranou spojení. Princip klouzavého okna je znázorněn na obrázku 3.4. Maximální aktuální velikost klouzavého okna je dána velikostí okna, kterou signalizuje druhá strana v TCP hlavičce. Pokud klouzavé okno dosáhne této velikosti, neměla by být po síti odeslána další data, dokud druhá strana nepotvrdí přijetí některých předchozích odeslaných dat. Přijetím potvrzení o přijetí odeslaných dat se tedy velikost klouzavého okna zmenšuje a naopak odesláním dalších dat po síti se velikost klouzavého okna zvětšuje.

Pokud jedna strana spojení přijme segment, ve kterém je aktuální velikost okna druhé strany deklarována jako nulová, nesmí dále odesílat žádná data. Pokud ale dále už nepřijme žádné potvrzení o přijetí odeslaných dat, které by indikovalo zvětšení okna, nebude vědět, kdy může opět začít data posílat. Řešením tohoto problému je nastavení časovače (persistent timer) na straně odesílatele v momentě, kdy přijme segment indikující nulovou velikost okna. Tento časovač pak periodicky posílá druhé straně malý paket, tzv. *WindowProbe*, na který druhá strana reaguje zasláním paketu s aktuální velikostí svého okna.



Obrázek 3.4: Princip klouzavého okna v TCP

Kapitola 4

DPDK

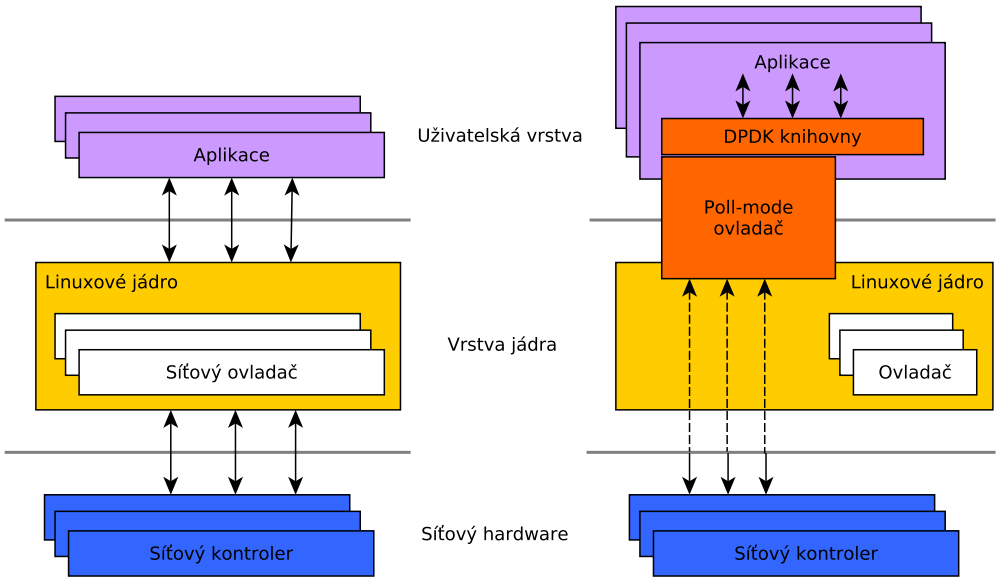
DPDK (Data Plane Development Kit) je framework obsahující sadu knihoven a ovladačů síťových karet určených pro rychlé zpracovávání paketů v uživatelské vrstvě systému. Mezi funkce frameworku patří podpora víceprocesového zpracování, poll-mode ovladačů, knihovny s paměťovými strukturami podporujícími přístup ke sdílené paměti bez nutnosti zámků, knihovny pro klasifikaci a plánování paketů, hash knihovna aj. Spojením funkcí těchto knihoven můžeme vytvořit komplexní aplikace pro rychlé zpracování paketů v uživatelské vrstvě systému [10].

V současnosti je DPDK plně kompatibilní s Linuxem a existuje také port na FreeBSD. DPDK se snaží o co nejefektivnější využití výkonu procesorů a síťových karet zejména společnosti Intel. Seznam podporovaných síťových karet jiných výrobců se ale neustále rozrůstá [15]. DPDK podporuje rovnoměrné rozložení provozu aplikací na všechna jádra procesoru, abychom dosáhli co největšího výkonu. Toho dosahuje využitím technologie RSS (Receive Side Scaling), která na síťové kartě počítá pro každý paket pomocí hashovací funkce hodnotu, podle které se pakety přidělují jednotlivým jádrům procesoru [20]. Hlavní výhodou DPDK je minimalizace režie linuxového jádra při komunikaci aplikace se síťovým hardwarem a jeho ovladači, jak je znázorněno na obrázku 4.1. Tímto postupem dosáhneme velké úspory výkonu při zpracovávání paketů.

4.1 Poll-mode ovladače

Linux většinou využívá k obsluze hardware ovladače založené na obsluze přerušení. Při příchodu paketu na síťovou kartu vyvolá ovladač přerušení. Jádro systému musí pozastavit svoji další aktivitu a zavolat obsluhu přerušení pro danou událost. V případě příchodu paketu je paket zkopírován do vstupní fronty jádra. Jádro se poté postará o zpracování paketu. Kód pro obsluhu přerušení má přednost před kódem zpracovávajícím paket. Při vysoké rychlosti příchozího toku paketů se může stát, že kód obsluhy přerušení bude stále dostávat přednost před zpracováním paketu. Vstupní fronta jádra se tak zaplní a systém je zahlcen. Tomuto problému se říká *receive-livelock* [1].

Framework DPDK využívá k obsluze síťových karet tzv. poll-mode ovladače, které běží v uživatelském prostoru systému. Tyto ovladače používají k obsluze síťových karet metodu pollingu. Ta spočívá v opakovaném aktivním dotazování síťové karty ohledně jejího stavu. Ovladač přistupuje k deskriptorům vstupních a výstupních front síťové karty bez jakýchkoliv přerušení systému, což umožňuje rychlé přijímání, zpracování a odesílání paketů aplikací v uživatelském prostoru za cenu nepřetržitého vytížení jádra procesoru. [14].



Obrázek 4.1: Zmenšení režie zpracování paketu s frameworkem DPDK. Převzato z [9].

4.2 Velké paměťové stránky

Kdykoliv proces používá operační paměť, procesor musí vyznačit část RAM, kterou tento proces využívá. Kvůli efektivitě je paměť RAM alokována po úsecích určité velikosti, kterým se říká stránky. Tyto stránky mají zpravidla 4096 B. Operační systém si musí pamatovat, které stránky patří kterému procesu a kde v paměti jsou uložené. Při požadavku procesu na přístup ke svým datům je u menšího počtu větších stránek vetší pravděpodobnost, že překlad virtuální na fyzickou stránku je udržován v paměti cache a přístup k ní je tedy mnohem rychlejší. Čím více stránek procesy alokují, tím je větší pravděpodobnost, že překlad stránky nebude uložen v paměti cache, což může mít vliv na výkon aplikace.

Mnoho současných architektur podporuje alokování stránek o větší velikosti než obvyklých 4096 B. Těmto paměťovým stránkám se říká různě podle systému – *huge pages* v Linuxu, *super pages* v BSD nebo *large pages* ve Windows [27]. Větší velikost paměťových stránek znamená jejich menší počet a větší velikost souvislých bloků paměti, což zvyšuje pravděpodobnost uchování překladu virtuálních adres na fyzické stránky v paměti cache pro konkrétní proces. Framework DPDK je přizpůsoben pro použití velkých paměťových stránek pro alokaci paměťových struktur používaných pro ukládání paketů do mezipaměti. Umí pracovat i bez nich, ale běh aplikací využívajících takto DPDK je poté neoptimální a občas i nestabilní. Velikosti velkých paměťových stránek jsou závislé na architektuře procesoru a podpoře operačního systému. Pro architekturu *x86_64* jsou například nejčastější velikosti 2 MB a 1 GB [11].

4.3 EAL (Environment Abstraction Layer)

EAL je vrstva zodpovědná za přístup DPDK aplikací k nízkoúrovňovým zdrojům. Poskytuje aplikacím jednotné rozhraní, které skrývá specifika daného prostředí a zajišťuje inicializaci zdrojů jako paměťový prostor, PCIe zařízení, časovače atd. EAL rovněž zajišťuje rozdělení

běhu aplikace mezi jednotlivá jádra procesoru. Informace o PCI zařízeních a paměťovém prostoru získává EAL z rozhraní jádra */sys* a z modulů jádra *uio_pci_generic*, *igb_uio* nebo *vfio-pci* [13].

Pro inicializaci jednotlivých vláken DPDK aplikace je použita linuxová knihovna *pthread*. Při spuštění aplikace specifikujeme, která jádra procesoru chceme využít pro běh aplikace pomocí parametru *-c COREMASK*, kde *COREMASK* je hexadecimální bitová maska jader procesoru. Po spuštění aplikace se nejprve spustí funkce *main()*. V ní by měla být volána funkce *rte_eal_init()*, která provede inicializaci jader přidělených pro běh aplikace v masce jader procesoru. Tato inicializace je provedena voláními funkcí dostupné knihovny pro správu vláken (např. u POSIX systémů knihovna *pthread*). Pokud není nastaveno jinak, je každému jádru procesoru přiděleno jedno vlákno.

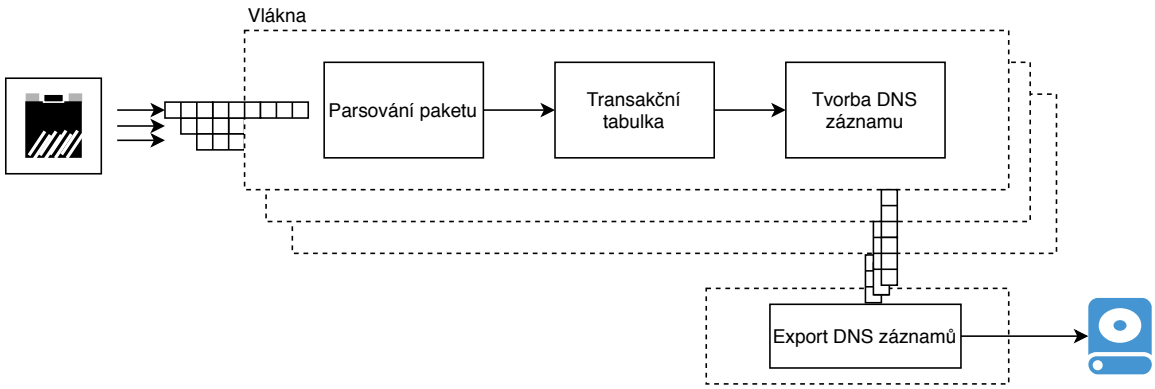
EAL již v inicializační fázi aplikace provede alokaci fyzické paměti aplikace pomocí funkce *mmap()* a velkých paměťových stránek. Alokace probíhá s využitím souborového systému linuxového jádra *hugetlbfs*. Tato paměť je poté přístupná knihovnám DPDK pro práci s pamětí, které si v ní mohou rezervovat pojmenované úseky pro svou potřebu [12].

Kapitola 5

DPDK DNS sonda

Správci domén prvního řádu, jako například správce *.cz* domény CZ.NIC, musejí zaručit nepřetržitý provoz svých doménových serverů. Doménové servery na takovéto úrovni musejí být schopny obsloužit desetitisíce až statisíce dotazů za sekundu [6]. Důležitým prvkem správy takových doménových serverů je podrobné monitorování veškerého provozu, který servery obsluhují. Kvalitní monitorování provozu může správce včas upozornit na potíže se serverem nebo potenciální útok na servery. K takovému monitorování slouží aplikace DPDK DNS sonda, která je vyvíjena na Fakultě informačních technologií Vysokého učení technického v Brně. Tato aplikace cílí na poskytnutí spolehlivého a podrobného monitorování DNS provozu na linkách o rychlosti až 10 Gbps.

Architektura aplikace je založena na vícevláknovém zpracování paketů, které umožňuje plně škálovat objem zpracovaných paketů podle počtu jader procesoru. Základní architektura aplikace je zobrazena na obrázku 5.1. Příchozí pakety jsou ze síťové karty rovnoměrně rozděleny do několika vstupních front. Každá vstupní fronta je navázána na jedno pracovní vlákno aplikace. Pracovní vlákno provede zpracování paketu a vytvoření příslušného záznamu o transakci dotaz-odpověď systému DNS. Paket je poté přeposlán dále po síti. Vytvořené záznamy jsou poté sbírány z pracovních vláken do jednoho exportního vlákna pomocí bezzámkové fronty. Exportní vlákno v současné době provádí export statistik provozu na disk ve formátech Parquet nebo PCAP. Parquet je komprimovaný sloupcový formát navržený pro efektivní uložení a procházení velkých strukturovaných dat. Je založen na principech představených společností Google v článku Dremel [24], který rozebírá efektivní přístup k velkým strukturovaným datům v reálném čase.



Obrázek 5.1: Architektura aplikace DPDK DNS sonda

Aplikace využívá možností frameworku DPDK představeného v předchozí kapitole. Paměť potřebná pro běh aplikace je předalokována ve velkých paměťových stránkách. K běhu aplikace na lince 10 Gbps je potřeba alokovat přibližně 2,4 GB velkých paměťových stránek. Pro rovnoměrné rozdělení příchozích paketů mezi pracovní vlákna aplikace je využito technologie RSS (Receive Side Scaling) [20]. Tato technologie spočítá pomocí hashovací funkce pro každý paket hodnotu, podle které dojde k přidělení paketu do příslušné vstupní fronty. Hashovací funkce danou hodnotu spočítá dle vybraných polí příchozího paketu. Konkrétně dle pětice zdrojová a cílová IP adresa, zdrojový a cílový port a identifikátor transportního protokolu. Aby bylo možné spárovat transakci dotaz-odpověď v jeden záznam, je nutné zajistit mapování opačných toků paketů na stejné pracovní vlákno. Toho je docíleno použitím speciálního vstupního klíče hashovací funkce [32].

Pracovní vlákno aplikace provádí tři hlavní operace – parsování paketu, párování dvojic dotaz-odpověď v transakční tabulce a vytváření záznamů o transakcích pro export. Parsování paketu pro vrstvy L2 až L4 je prováděno manuálně procházením paketu pomocí offsetů. Pro parsování DNS hlavičky a dat je použita knihovna *libknot* od správce *.cz* domény CZ.NIC. Z informací extrahovaných z paketu je vytvořen záznam, který je buď vložen do transakční tabulky, nebo je spárován s jiným záznamem v transakční tabulce. Transakční tabulka je rychlá hashovací tabulka unikátní pro každé pracovní vlákno, která udržuje dosud nespárované záznamy transakcí dotaz-odpověď. Spárování dvojic dotaz-odpověď probíhá pomocí stejné pětice hodnot, kterou používá RSS a navíc jsou porovnány hodnoty ID z DNS hlavičky paketu a volitelně i doménové jméno QNAME ze sekce dotazu v datové části paketu. Pokud je nový záznam spárován s nějakým záznamem v transakční tabulce, jsou tyto dva záznamy sloučeny do jednoho a tento záznam je transformován do vybraného exportního formátu (Parquet nebo PCAP). Vytvořený záznam připravený pro export je poté vložen do bezzámkové fronty, která spojuje pracovní vlákna s exportním vláknem.

Exportní vlákno zapisuje připravené záznamy do souboru na disku. Pro zlepšení výkonu zápis neprobíhá po jednotlivých záznamech, ale po dávkách o několika milionech záznamů. Exportní vlákno také dokáže omezovat exportní soubory podle zvolené maximální velikosti nebo stáří souboru.

Kapitola 6

Rozšíření aplikace DPDK DNS sonda

Jedním z úkolů této práce je optimalizovat výkon aplikace DPDK DNS sonda, aby dokázala monitorovat linku o rychlosti 10 Gbps. Prvním krokem tohoto úkolu tedy bylo identifikovat části aplikace, které nejvíce brzdí zpracování paketu. K tomuto účelu byla provedena měření rychlosti aplikace v různých fázích zpracování paketu. Jako testovací prostředí posloužila síťová laboratoř L311 na Fakultě informačních technologií Vysokého učení technického v Brně. Síťový provoz DNS paketů na lince 10 Gbps byl nasimulován pomocí generátoru síťového provozu Spirent TestCenter SPT-2000A. Toto zařízení nejenže generuje síťový provoz o rychlosti 10 Gbps, ale umožňuje také monitorovat statistiky příchozího provozu. Aplikace DPDK DNS sonda byla spuštěna na serveru s frameworkem DPDK ve verzi 18.05. Hardwarová konfigurace serveru je následující:

- **CPU:** Intel Xeon E5-2620@2.00 GHz
- **RAM:** 32 GB DDR3@1600 MHz
- **Operační systém:** Debian 8.0 "Jessie", jádro Linux 4.9.13
- **NIC:** Intel X-520-2, 2x 10 Gb

6.1 Výkonnostní testování aplikace DPDK DNS sonda

Aby bylo možné přibližně odhadnout část aplikace, kde dochází k největšímu zpomalení, byla aplikace spouštěna postupně jen s některými částmi procesu zpracování paketu. Každý typ měření byl poté proveden na různých počtech pracovních vláken, aby bylo ověřeno správné škálování aplikace. Výsledky měření jsou zobrazeny v tabulce 6.1.

Nejdříve byla aplikace spuštěna pouze v režimu jednoduchého přeposílání paketů, aby se ověřila základní funkčnost a rychlost aplikace. V tomto režimu dokázalo i jedno pracovní vlákno zpracovávat plnou linku o rychlosti 10 Gbps. Dále bylo k přeposílání paketů přidáno i jejich parsování. Zde výkon aplikace dramaticky poklesl, jak je vidět v tabulce 6.1. Bylo tedy provedeno profilování aplikace pomocí programu *gprof*. To ukázalo, že největší čas stráví aplikace dynamickou alokací paměti. Po přezkoumání kódu byla tato alokace nalezena v knihovně *libknot* při parsování datové části paketu DNS. Jelikož pro exportované statistiky jsou z většiny potřeba jen údaje z DNS hlavičky paketu, bylo vypnuto parsování datové části paketu DNS a měření bylo provedeno znovu. I když výkon aplikace oproti jednoduchému

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
L2-FWD	10	10	10	10	10	10	10	10
DNS parsing s EDNS	1,4	2,7	4	5,2	6,4	7	7	7
DNS parsing bez EDNS	5,3	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka	4	8	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka + Parquet export	0,9	1,8	2,7	3,5	4,4	4,6	4,8	5

Tabulka 6.1: Počet průchozích DNS paketů za sekundu při rychlosti linky 10 Gbps

přeposílání paketů klesl, byl tento pokles v očekávatelných mezích a aplikace velmi dobře škálovala.

Byla tedy provedena další měření, kde byly postupně přidávány jednotlivé fáze aplikace – párování záznamů v transakční tabulce a tvorba exportních záznamů v Parquet formátu. Zde se jako největší přítěž projevila knihovna pro export dat v Parquet formátu. Tento nedostatek je způsoben tím, že Parquet je relativně nový komplexní formát pro ukládání strukturovaných dat, pro který v současnosti existuje pouze jedna knihovna v C++ od tvůrce tohoto formátu, skupiny Apache [18].

6.2 Optimalizace parsování DNS

Největší dopad na výkon aplikace DPDK DNS sonda má dle měření výše parsování datové části paketu DNS. Aplikace v současnosti používá pro parsování DNS hlavičky i datové části paketu knihovnu *libknot*. V kapitole 2.3.1 byl popsán formát záznamů DNS, které jsou obsahem datové části paketu DNS. Knihovna *libknot* tyto záznamy prochází jeden po druhém a pro každý dynamicky alokuje paměť, do které ukládá informace z daného záznamu.

Při bližším pohledu zjistíme, že většina statistik sbíraných aplikací je přítomna v DNS hlavičce paketu, případně hned v první sekci datové části – v sekci dotazu. Tyto části jsou v paketu vždy pevně za sebou a je tedy možné je jednoduše získat lineárním procházením paketu. Tento úkon je relativně jednoduchý na implementaci a není tedy potřeba využívat externí knihovnu. Jediné položky, které jsou potřeba získat z ostatních sekcí datové části DNS paketu, jsou položky záznamu EDNS. Ten se nachází v poslední sekci datové části DNS paketu – sekci s doplňujícími informacemi. EDNS záznamy obsahují položky rozšiřující původní DNS standard o nové informace k přenosu. Definovány jsou v RFC 6891 [7] a RFC 7871 [3].

Pro ostatní sekce datové části paketu DNS tedy není potřeba dynamicky alokovat dodatečnou paměť pro uložení extrahovaných informací. Záznam EDNS je identifikován tak, že položka NAME, kterou každý záznam začíná, je prázdná. I když v hlavičce paketu DNS jsou přesné informace o počtu jednotlivých záznamů v každé sekci datové části paketu, nelze tyto záznamy jednoduše přeskočit. Položky NAME a RDATA v každém záznamu mají proměnnou velikost, je tedy nutné pro každý záznam provést alespoň základní zpracování.

S výše uvedeným postupem je možné implementovat vlastní parsování DNS paketu, které nevyžaduje dynamickou alokaci paměti. Je tedy možné se obejít bez knihovny *libknot*, což by mělo výrazně urychlit zpracování paketů v pracovních vláknech aplikace.

Kapitola 7

Implementace optimalizovaného parsování DNS

Optimalizace parsování DNS spočívá v nahrazení funkcí knihovny *libknot* pro parsování DNS paketů vlastní implementací. Funkce, které bylo potřeba nahradit, je možné rozdělit na dvě hlavní části – funkce pro parsování DNS hlavičky a funkce pro parsování datové části DNS paketu.

V příloze B je zobrazen seznam položek exportovaných aplikací DPDK DNS sonda. Položky *ID*, *Qname*, *Domainname* a poté položky *AA* a dále jsou získávány z DNS části paketu. Z těchto položek se pouze *Qname*, *Domainname* a poté *QType* a dále získávají z datové části paketu. Ostatní položky se získávají již z DNS hlavičky, která má pevnou velikost. V datové části paketu se pak položky *Qname*, *Domainname*, *QType* a *QClass* získávají z prvního záznamu sekce dotazů, která následuje bezprostředně za DNS hlavičkou. Většina exportovaných položek se tedy získává na začátku paketu. Pouze položky získané z EDNS záznamu se nacházejí až ke konci datové části paketu. EDNS záznam se totiž nachází v sekci s doplňujícími informacemi, která je v datové části paketu až poslední. Nelze tedy zcela přeskočit parsování datové části DNS paketu, ale je možné ho oproti funkcím knihovny *libknot*, které plně parsují všechny záznamy datové části paketu, výrazně zjednodušit za účelem optimálního výkonu aplikace.

7.1 Parsování DNS hlavičky

Formát DNS hlavičky je zobrazen v kapitole 2.3 na obrázku 2.2. Při vstupu do funkce parsující DNS část paketu dostaneme ukazatel na začátek DNS hlavičky daného paketu. Jelikož má DNS hlavička pevnou velikost 12 bytů, je možné jednotlivé její položky získat vymaskováním daných offsetů od jejího začátku.

Hned z prvních dvou bytů DNS hlavičky získáme ID, které jednoznačně identifikuje danou dvojici dotaz-odpověď. Následující 2 byty obsahují různé kontrolní bity a chybové kódy sloužící ke kontrole korektního průběhu dané transakce dotaz-odpověď. Zbylých 8 bytů hlavičky slouží pro čtyři dvoubytové čítače, které určují počet záznamů v jednotlivých sekcích datové části DNS paketu. Tyto čítače nejsou potřeba pouze pro naplnění položek statistik, které aplikace exportuje, ale jsou nutné i pro následné parsování datové části DNS paketu.

7.2 Parsování datové části DNS paketu

První sekcí v datové části paketu je sekce s dotazy. Formát záznamů v této sekci je zobrazen v kapitole 2.3.1 na obrázku 2.4. Záznam se skládá ze tří polí, přičemž první pole obsahující doménového jméno má variabilní délku. Z hlediska exportovaných položek aplikaci zajímají všechna pole prvního záznamu v této sekci. Tento záznam je tedy nutné kompletně zpracovat.

Pole QTYPE a QCLASS jsou pevné dvoubytové položky, které lze získat vyčtením podle offsetu. Nejdříve je ale nutné zpracovat pole s doménovým jménem. Formát doménového jména při paketovém přenosu je popsán v RFC 1035 [26]. Doménové jméno se v tomto případě skládá z domén jednotlivých úrovní, které se skládají z jednoho bytu určujícího délku následující domény v bytech a bytů s názvem domény. Konec doménového jména je určen bytem s délkou následující domény rovnou nule. Doménové jméno je naplněno do exportované položky *Qname* v klasickém textovém formátu, kde jsou jednotlivé domény odděleny tečkou.

Ve zbylých záznamech všech ostatních sekcí v datové části DNS paketu aplikaci již z hlediska exportovaných položek zajímá pouze EDNS záznam v poslední sekci s doplňujícími informacemi. Formát záznamů v ostatních sekcích je zobrazen v sekci 2.3.1 na obrázku 2.3. Na tomto obrázku lze vidět, že záznamy mají na rozdíl od záznamů v sekci dotazů dvě pole variabilní délky – NAME a RDATA. Pole NAME je doménové jméno, které lze rozparsovat stejně jako v sekci dotazů. Záznam EDNS identifikujeme tak, že jeho doménové jméno je prázdné, tedy skládá se pouze z jednoho bytu určujícího délku další domény a tento byte má hodnotu nula.

Jediným rozdílem položky NAME oproti QNAME je, že může využít komprese užívané pro zmenšení datové části DNS paketu. Tato komprese spočívá v tom, že pokud se některá doménová jména nebo jejich části v paketu opakují, můžeme místo jejich opětovného zapsání do pole NAME použít odkaz na místo v paketu, kde se již tato část doménového jména vyskytovala. Tento odkaz je v poli NAME určen speciální hodnotou v bytu s délkou následující domény. Protože obsah doménových jmen v těchto záznamech již nepotřebujeme pro export, při parsování pouze přeskočíme tento jeden délkový byte. Následující pole záznamu mají pevnou délku až na poslední pole RDATA. Jeho délka je ovšem určena hodnotou předcházejícího pole RDLENGTH, je tedy snadné toto pole při parsování rychle přeskočit. Tímto způsobem je možné rychle přeskakovat ostatní záznamy v datové části DNS paketu dokud se nenarazí na EDNS záznam nebo na konec paketu.

Pokud je nalezen v datové části DNS paketu EDNS záznam, jsou jeho obsahem naplněny příslušné položky pro export statistik. Dle RFC 6891 [7] mají některá pole EDNS záznamu jiný význam než u ostatních typů záznamů. Položka *EdnsUDP*, která stanovuje maximální možnou velikost UDP dat přenositelnou v jednom paketu přes síť, je naplněna obsahem pole CLASS. Položky *EdnsDO* a *EdnsVersion* jsou naplněny vymaskováním určitých bitů pole TTL. Zbylé exportované položky vztahující se k EDNS záznamu jsou plněny z pole RDATA. To se skládá z libovolného počtu EDNS option položek, jejichž formát je popsán v subsekci 2.3.2.

Kapitola 8

Testování optimalizovaného parsování DNS

Optimalizované parsování DNS bylo testováno ve dvou fázích – testy korektnosti a výkonnostní testy. Testy korektnosti spočívaly ve spuštění aplikace DPDK DNS sonda nad vzorovými PCAP soubory s různými typy DNS provozu. Testován byl jak základní provoz bez EDNS záznamů, tak provoz s různými druhy polí v EDNS záznamech. Nad každým PCAP souborem byla spuštěna aplikace s parsováním pomocí knihovny *libknot* a poté s optimalizovaným manuálním parsováním. Vyexportované soubory byly poté porovnány a bylo ověřeno, že obě varianty aplikace vyexportovaly stejné statistiky provozu.

Pro výkonnostní testy posloužila stejně jako při úvodním měření v kapitole 6 síťová laboratoř L311 na Fakultě informačních technologií Vysokého učení technického v Brně. Síťový provoz DNS paketů na lince 10 Gbps byl opět simulován pomocí generátoru síťového provozu Spirent TestCenter SPT-2000A. Aplikace DPDK DNS sonda bylo opět spuštěna na serveru s nainstalovaným frameworkem DPDK verze 18.05 s následující hardwarovou konfigurací:

- **CPU:** Intel Xeon E5-2620@2.00 GHz
- **RAM:** 32 GB DDR3@1600 MHz
- **Operační systém:** Debian 8.0 "Jessie", jádro Linux 4.9.28-2
- **NIC:** Intel X-520-2, 2x 10 Gb

Protože aplikace DPDK DNS sonda je stále ve vývoji, byla všechna měření z kapitoly 6.1 provedena znovu na aktuální verzi aplikace jak s parsováním pomocí knihovny *libknot*, tak s optimalizovaným parsováním DNS.

8.1 Testování na 33 % EDNS provozu

Prvním testovaným scénářem na 10 Gbps lince byla simulace reálného DNS provozu. Tento provoz byl simulován tak, že na 10 Gbps lince pouze 33 % DNS provozu obsahovalo EDNS záznamy. EDNS záznamy jsou rozšířením oproti původní specifikaci protokolu DNS a jsou používány pouze v určitých případech pro přenos dodatečných informací přes protokol DNS. V reálném provozu EDNS záznamy obsahuje pouze část dotazů a odpovědí, která byla pro účely těchto výkonnostních testů odhadnuta na jednu třetinu celkového provozu.

Stejně jako v sekci 6.1 byly obě verze aplikace postupně spouštěny jen s některými částmi procesu zpracování paketu. Výsledky testů jsou zobrazeny pro verzi s parsováním pomocí knihovny *libknot* v tabulce 8.1 a pro verzi s optimalizovaným parsováním v tabulce 8.2. Měření pro DNS parsing bez EDNS a DNS parsing s EDNS se liší tím, jestli má aplikace zapnuté prohledávání paketu na EDNS záznam. Pokud má aplikace tuto funkci vypnutou, i když přijde paket obsahující EDNS záznam, aplikace ho nevyhledá a neexportuje příslušné položky statistik.

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS	6,2	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka	4,7	9,4	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka + Parquet export	1,3	2,5	3,7	4,8	6	6,8	6,8	6,8	6,8	6,8
DNS parsing s EDNS	1,4	2,75	4,05	5,2	6,45	7,07	7,1	7,03	6,85	7,05
DNS parsing s EDNS + Transakční tabulka	1,35	2,6	3,8	4,9	6,05	6,7	6,75	6,75	6,65	6,75
DNS parsing s EDNS + Transakční tabulka + Parquet export	0,7	1,35	2	2,6	3,15	3,5	3,6	3,6	3,7	3,7

Tabulka 8.1: Parsování s knihovnou *libknot* při 33 % EDNS provozu na 10 Gbps lince

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS	9,9	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka	6,9	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka + Parquet export	1,4	2,8	4,2	5,4	6,7	7,6	7,55	7,3	7,4	7,4
DNS parsing s EDNS	7,15	10	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka	4,9	9	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka + Parquet export	1,3	2,4	3,6	4,5	5,7	6,1	6,4	6,5	6,5	6,7

Tabulka 8.2: Manuální parsování při 33 % EDNS provozu na 10 Gbps lince

Z tabulek je patrné, že při použití optimalizovaného parsování DNS došlo ke znatelnému nárůstu výkonu aplikace. I při parsování bez EDNS došlo k nárůstu výkonu na jednom jádře z 6,2 Gbps na 9,9 Gbps. Při parsování s EDNS výkon na jednom jádře narostl dokonce z 1,4 Gbps na 7,15 Gbps. Měření s dalšími částmi procesu zpracování paketu ukazují, že momentálně největší dopad na výkon má stále export statistik v Parquet formátu. Přesto výkon na jednom jádře narostl z 0,7 Gbps na 1,3 Gbps, což je skoro dvojnásobné zrychlení. Měření s více pracovními jádry ukazují konzistentní škálování a tedy i konzistentní nárůst výkonu optimalizované aplikace oproti aplikaci používající knihovnu *libknot*. Škálovaný výkon aplikace narazí na horní limit při asi šesti pracovních jádrech, což je způsobeno procesorem serveru, na kterém byla měření prováděna. Tento procesor má 6 fyzických a 12 logických jader, čehož je dosaženo technologií HyperThreading, kdy jedno fyzické jádro sdílí dvě logická jádra. Aplikace DPDK DNS sonda ovšem všechna pracovní jádra naplno vytěžuje, a tak se ztrácí výkonnostní výhoda této technologie.

8.2 Testování na 100 % EDNS provozu

Druhým testovaným scénářem na 10 Gbps lince bylo plné vytížení linky DNS provozem obsahujícím záznamy EDNS. Výsledky těchto měření jsou zobrazeny v tabulkách 8.3 a 8.4. Měření odhalila, že nárůst výkonu aplikace s optimalizovaným parsováním DNS oproti verzi s knihovnou *libknot* přibližně odpovídá nárůstu výkonu při předchozím měření s 33 % DNS provozu.

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka	7,5	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka + Parquet export	2,1	4	6,1	8,1	9,4	9,3	9	9	9,5	9,8
DNS parsing s EDNS	2	3,6	5,2	6,5	7,9	8,5	8,2	8,1	8,2	8,7
DNS parsing s EDNS + Transakční tabulka	1,85	3,4	5	6,1	7,5	8,3	8,1	7,9	8,05	8,55
DNS parsing s EDNS + Transakční tabulka + Parquet export	1	1,9	2,8	3,6	4,3	4,9	5	5,1	5,25	5,2

Tabulka 8.3: Parsování s knihovnou *libknot* při 100 % EDNS provozu na 10 Gbps lince

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka	10	10	10	10	10	10	10	10	10	10
DNS parsing bez EDNS + Transakční tabulka + Parquet export	2,3	4,5	6,7	8,8	9,8	9,6	9,5	9,6	9,8	9,9
DNS parsing s EDNS	7,8	10	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka	5,6	9,7	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka + Parquet export	1,8	3,4	4,8	5,9	7,4	8,1	8	8	8	9

Tabulka 8.4: Manuální parsování při 100 % EDNS provozu na 10 Gbps lince

U obou verzí aplikace došlo při tomto měření k nárůstu výkonu oproti scénáři s 33 % EDNS provozu. Tento nárůst lze vysvětlit tak, že pakety obsahující EDNS záznamy (zejména odpovědi serveru) mají až několikanásobně větší velikost oproti paketům bez EDNS záznamů. To znamená, že na 10 Gbps lince těchto paketů za sekundu projde podstatně méně a aplikace tedy musí zpracovávat méně paketů. Dodatečný výpočetní čas potřebný k prohledávání větších paketů na EDNS záznamy pak není tak vysoký, aby se negativně projevil i při menší paketové zátěži a dojde tedy k nárůstu výkonu.

Kapitola 9

Zpracování DNS přes TCP

Aplikace DPDK DNS sonda podporuje export statistik DNS provozu pouze při přenosu přes transportní protokol UDP. Standard RFC 1035 [26] ale jasně definuje i možnost použití transportního protokolu TCP. K tomu většinou dochází, pokud je daná DNS zpráva příliš velká pro přenos přes UDP, které má standardem daný limit velikosti DNS zprávy 512 bytů. Některé typy dotazů mají dokonce standardem nařízeno použití transportního protokolu TCP. Příkladem může být přenos zónových souborů mezi jednotlivými DNS servery, který je uskutečněn pomocí dotazu typu AXFR [21]. Velké zónové soubory mohou dosahovat velikostí v řádech megabytů až gigabytů, a tak ani není možné je přenést pomocí protokolu UDP.

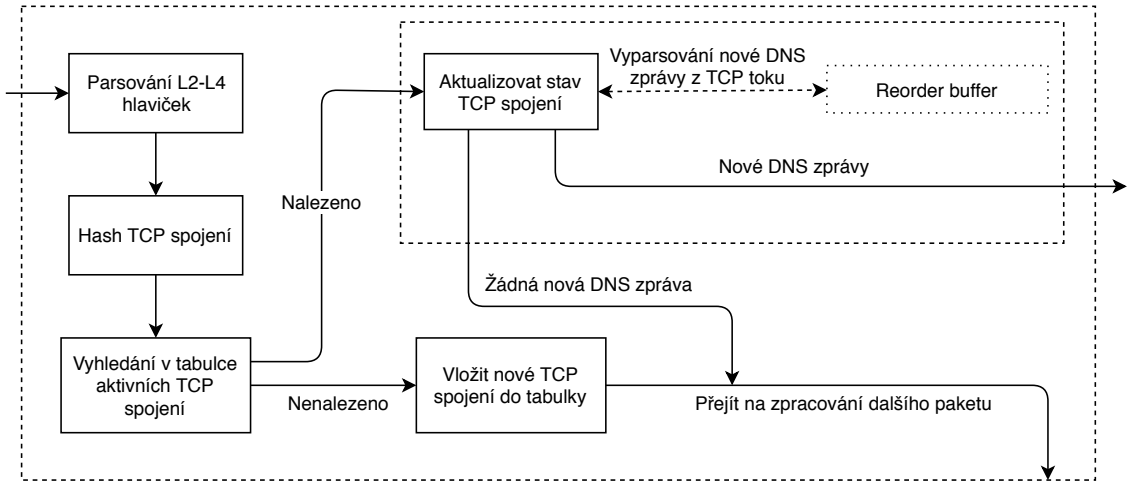
V reálném provozu dosahuje přenos DNS přes TCP naprosto minimálního podílu na celkovém provozu. Dle statistik doménových serverů firmy CZ.NIC, které zajišťují obsluhu *.cz* domény, tvoří provoz DNS přes TCP ani ne jedno procento celkového provozu na *.cz* doméně [5]. Dle diskuzí s administrátory z firmy CZ.NIC by se ale v následujících letech měl projevit vzestupný trend v používání TCP pro přenos DNS. S navyšováním dostupné šířky pásma internetové sítě pro širokou veřejnost ztrácí na důležitosti jednoduchost protokolu UDP a naopak nabývá na významu spolehlivost a bezpečnost protokolu TCP. Pokud má být aplikace DPDK DNS sonda použita pro spolehlivý export statistik DNS provozu, je nutné podporovat i přenos přes protokol TCP.

Zpracování provozu využívajícího transportní protokol TCP je mnohem komplexnější než u protokolu UDP. Při přenosu DNS přes UDP se v jednom paketu vždy přenáší jedna celá DNS zpráva. Protokol UDP také nemá žádný mechanismus pro kontrolu spolehlivého doručení paketů. Nedochází tedy k opětovném zasílání stejných paketů a není ani třeba řešit doručení paketů v pořadí, ve kterém byly odeslány. Aplikace DPDK DNS sonda tedy může zpracovávat pakety po jednom tak, jak přicházejí, aniž by bylo třeba udržovat jakékoliv stavové informace o spojení mezi klientem a serverem.

Protokol TCP ovšem na rozdíl od UDP implementuje mechanismy pro spolehlivé doručení dat. Spojení přes protokol TCP je kvůli tomu stavové a obě strany spojení si musejí udržovat informace pro každé navázané spojení. Protokol TCP dále nezaručuje zaslání jedné celé DNS zprávy v jednom paketu. Uživatelské aplikace zapisují jednotlivé DNS zprávy do soketu. TCP tato data ovšem nevidí jako jednotlivé DNS zprávy ale jako souvislý proud bytů bez jakéhokoliv konkrétního významu. Z tohoto souvislého proudu bytů poté TCP podle potřeby sestavuje jednotlivé segmenty tak, aby každý paket naplnilo co nejvíce daty. Jedna velká DNS zpráva tedy může být poslána po síti jako několik paketů a naopak několik malých DNS zpráv může být odesláno jako jeden paket. Aby bylo možné na straně příjemce od sebe rozlišit jednotlivé DNS zprávy v TCP toku, definuje standard RFC 1035

[26], že při přenosu DNS přes TCP předchází každé DNS zprávě bezprostředně před DNS hlavičkou 2-bytové pole, jehož hodnota určuje velikost nadcházející DNS zprávy. Pro spolehlivé zpracování DNS zpráv v TCP toku je tedy nutné sledovat celé TCP spojení od jeho počátku, aby byla zachycena již první DNS zpráva v daném TCP spojení, podle které je poté možné identifikovat i počátky dalších DNS zpráv přenesených v daném TCP spojení. Pro zajištění spolehlivého doručení dat je TCP také schopné rekonstruovat datový tok i při příchodu paketů mimo pořadí, případně je schopné opětovně zaslat pakety, které se po cestě k příjemci ztratily.

Aplikace DPDK DNS sonda tedy v případě přenosu DNS před TCP nemůže naivně zpracovávat pakety po jednom tak, jak přicházejí, ale musí implementovat mechanismy pro udržování stavu TCP spojení a pro rekonstrukci datového toku. V kapitole 5 na obrázku 5.1 byla představena architektura aplikace DPDK DNS sonda. Tato architektura je postavena na zpracování DNS zpráv po jednotlivých paketech. Aby bylo možné zpracovávat DNS přes TCP, je nutné rozšířit modul *Parsování paketu*. Pro pakety využívající protokol UDP tento modul zpracuje hlavičky L2-L4 a poté zpracuje DNS hlavičku a datovou část paketu. Získanými informacemi naplní jeden DNS záznam a pošle ho dále do modulu *Transakční tabulka*, který se pokusí o spárování dvojice dotaz-odpověď.



Obrázek 9.1: Návrh zpracování DNS nad TCP

Návrh modulu *Parsování paketu* pro DNS přes TCP je představen na obrázku 9.1. Pro každý paket proběhne zpracování L2-L4 hlaviček stejně jako při zpracování DNS přes UDP. K udržování stavu jednotlivých TCP spojení bude sloužit tabulka aktivních TCP spojení. Tato tabulka bude implementována jako hash tabulka, přičemž klíčem k jednotlivým položkám bude hash vypočítaný z čtveřice hodnot – zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port. Tato čtveřice bude unikátně identifikovat každé TCP spojení. Může se stát, že identickou čtveřici bude mít i paket odeslaný přes protokol UDP, ale protokol UDP bude identifikován již během zpracování L2-L4 hlaviček a takový paket tedy nebude vůbec porovnáván oproti tabulce aktivních TCP spojení. Pokud daný paket nebude odpovídat žádnému spojení v tabulce aktivních TCP spojení, bude v tabulce vytvořen nový záznam. Jelikož první paket v každém TCP spojení je SYN paket začínající proceduru navazování spojení (three-way handshake), nebude z něj vyextrahována nová DNS zpráva a aplikace tedy může přejít na zpracování dalšího paketu.

Pokud bude zpracovávaný paket spárován s existujícím spojením v tabulce aktivních TCP spojení, bude stav tohoto spojení aktualizován podle obsahu zpracovávaného paketu. Pokud paket nemá v TCP hlavičce sekvenční číslo, které je v rámci spojení momentálně očekáváno, bude paket zařazen do bufferu. Tento buffer, který je unikátní pro každé TCP spojení, zajišťuje sestavení paketů TCP toku ve správném pořadí. Pokud je sekvenční číslo zpracovávaného paketu shodné se sekvenčním číslem nějakého paketu v bufferu, jedná se o opětovně zaslaný paket, který není nutné dále zpracovávat. Pokud sekvenční číslo zpracovávaného paketu odpovídá dalšímu očekávanému sekvenčnímu číslu spojení, aplikace se pokusí z paketu extrahovat všechny DNS zprávy, které obsahuje. Tyto DNS zprávy jsou poté předány modulu *Transakční tabulka* na spárování dvojic dotaz-odpověď.

Kapitola 10

Implementace zpracování DNS přes TCP

Rozšíření aplikace DPDK DNS sonda o podporu zpracování DNS přes TCP vyžaduje implementaci tří hlavních částí – tabulky aktivních TCP spojení, konečného automatu pro udržování stavu spojení a bufferu pro rekonstrukci datového toku ve správném pořadí.

Pro implementaci tabulky aktivních TCP spojení bylo využito již existující implementace hashovací tabulky v aplikaci DPDK DNS sonda v souboru *TransactionTable.h*. Tato tabulka je využita také pro implementaci modulu *Transakční tabulka*, který páruje dvojice DNS zpráv dotaz-odpověď. Jedná se o hashovací tabulku, která umožňuje programátorovi zvolit, jakým způsobem bude počítána hodnota hashe a také jaká data budou v položkách tabulky uložena. Hodnota hashe je pro tabulku aktivních TCP spojení počítána dle čtveřice – zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port.

Jako data ukládaná v položce hashovací tabulky slouží objekty třídy `DnsTcpConnection` s následujícími atributy:

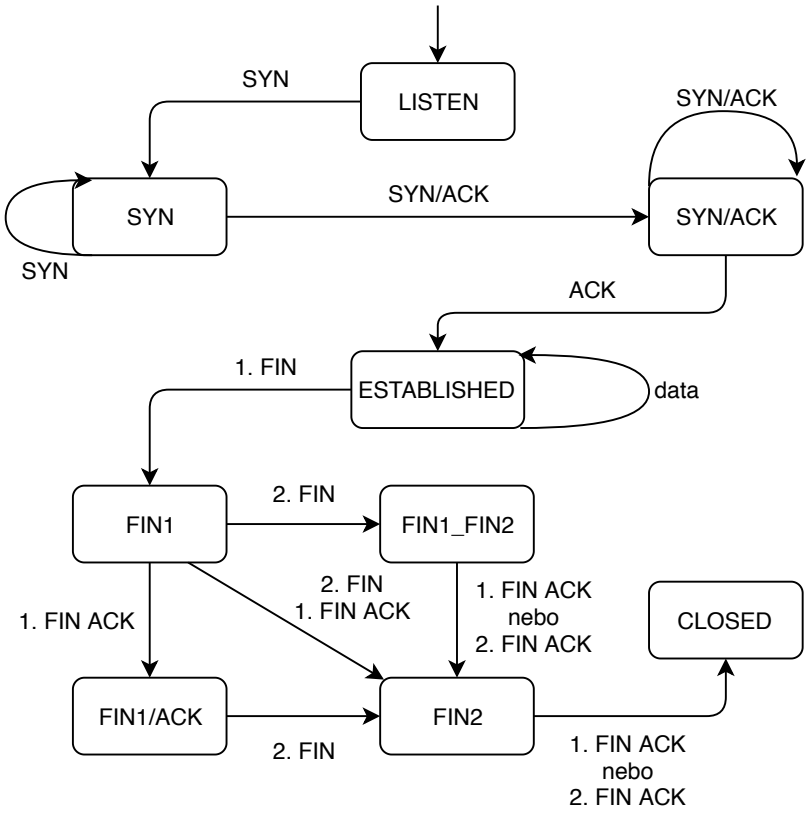
```
uint32_t m_hash;
uint32_t m_isn[2];
uint32_t m_next_seq[2];
bool m_fin[2];
TcpConnectionState m_state;
uint16_t m_unparsed_msg[2];
struct rte_mbuf* m_buffer_head[2];
struct rte_mbuf* m_buffer_tail[2];
```

Kromě hodnoty hashe jsou zde uloženy také hodnoty ISN a dalšího očekávaného sekvenčního čísla pro obě strany spojení. Dále se zde nachází informace o zachycení paketu s příznakem FIN pro obě strany spojení a atribut aktuálního stavu spojení. Každá strana spojení má také svůj vlastní buffer pro rekonstrukci datového toku. Pro oba buffery je zde uložen ukazatel na jejich první a poslední položku. Obě strany spojení pak mají svůj vlastní atribut `m_unparsed_msg`. Tento atribut slouží pro optimalizaci extrahování DNS zprávy, která je rozložena přes několik segmentů, z bufferu.

Při příchodu nového paketu, který patří k danému TCP spojení, dojde nejprve k aktualizaci atributu stavu spojení. Podle něj se poté aplikace může pokusit o získání nové DNS zprávy z nového paketu, případně z paketů v bufferu dané strany spojení, pokud nový paket v bufferu zaplnil mezeru v datovém toku.

10.1 Konečný automat TCP spojení

Každá strana TCP spojení si udržuje stavové informace o daném spojení. Stav spojení je na každé straně udržován nejen na základě odeslaných a přijatých paketů, ale také podle interního stavu dané strany spojení. Tyto stavové informace vycházejí z konečného automatu zobrazeného na obrázku 3.2 v sekci 3.2. Průchod tímto konečným automatem probíhá na každé straně spojení zvlášť. Aplikace DPDK DNS sonda se ale při spuštění nachází v síti někde mezi oběma konci spojení a nemá tedy přístup k interním stavovým informacím obou stran spojení. Stav spojení si aplikace tedy musí odvodit pouze dle zachycených paketů. Jelikož se aplikace nachází mezi oběma konci spojení, dává smysl, aby si pamatovala pouze jeden stav spojení pro oba konce spojení dohromady. Atribut stavu spojení aplikace aktualizuje pro každý zachycený paket daného spojení. Protože je tento atribut sdílený pro obě strany spojení, bylo potřeba modifikovat konečný automat TCP. Modifikovaný automat použitý v aplikaci DPDK DNS sonda je zobrazen na obrázku 10.1.



Obrázek 10.1: Konečný automat pro sledování stavu TCP spojení

Tento automat je schopný pouze podle analýzy zachycených paketů detekovat jak proceduru navazování spojení (three-way handshake), tak proceduru korektního ukončení spojení. Stavy LISTEN, SYN a SYN/ACK slouží k obsluze navázání spojení. Při použití mechanismu TCP Fast Open je možné už během této fáze posílat v paketech data [2]. Aplikace tedy už během této fáze kontroluje pakety na DNS zprávy a případně je zpracovává.

Hlavní zpracování datového toku TCP spojení probíhá v rámci stavu ESTABLISHED. V tomto stavu se spojení skládá převážně z paketů s DNS zprávami a ACK paketů potvr-

zujících přijetí dat druhou stranou. Během tohoto stavu také dochází k hlavnímu využití bufferu pro rekonstrukci datového toku ve správném pořadí.

Fáze ukončení spojení nastává, když jedna strana spojení pošle paket s nastaveným příznakem FIN. Tento paket způsobí přechod ze stavu ESTABLISHED do stavu FIN1. Pro korektní ukončení spojení se aplikace snaží detekovat zaslání paketu s příznakem FIN oběma stranami spojení a potvrzení přijetí těchto paketů oběma stranami. Výsledkem je postupný přechod do stavu CLOSED, který indikuje ukončení spojení a způsobí vymazání spojení z tabulky aktivních TCP spojení. Aplikace musí detekovat kompletní proceduru ukončení spojení, protože i když jedna strana spojení pošle paket s nastaveným příznakem FIN, druhá strana může mít ještě data k odeslání a tato data je dle standardu oprávněna poslat.

Speciálním případem ukončení spojení je zaslání paketu s nastaveným příznakem RST. Tímto příznakem dává daná strana spojení najevo, že s okamžitou platností ukončuje spojení. Tento paket může být zaslán v jakémkoliv stavu spojení a způsobí přechod do stavu CLOSED a vymazání spojení z tabulky aktivních TCP spojení. Pro zachování přehlednosti konečného automatu v něm tato eventualita není zakreslena, aplikace ji však implementuje.

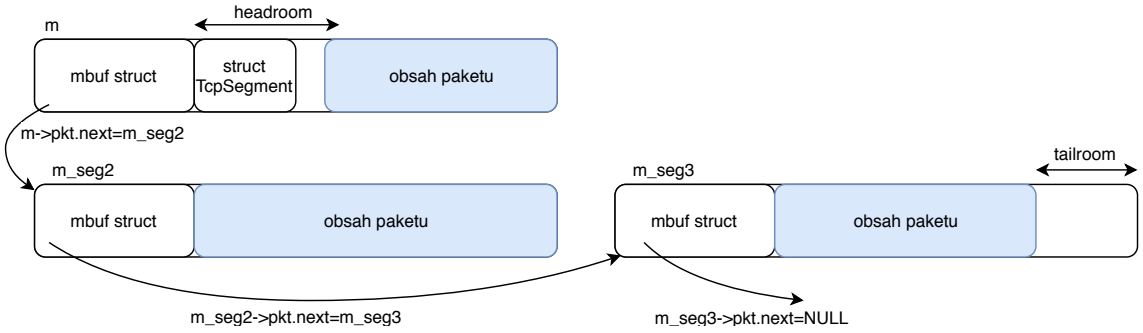
10.2 Přeuspořádací buffer

Pro získání DNS zpráv z TCP toku je potřeba zpracovávat pakety ve správném pořadí. K tomuto úkolu slouží přeuspořádací buffer. Každá strana TCP spojení má svůj vlastní buffer. Tento buffer je v aplikaci implementován jako obousměrně vázaný seznam. Ukazatele na první a poslední položku tohoto seznamu jsou uloženy v položkách tabulky aktivních TCP spojení jako atributy `m_buffer_head` a `m_buffer_tail`. Jednotlivé položky tohoto seznamu mají následující tvar:

```
struct TcpSegment {
    uint32_t seq_number;
    uint32_t segment_size;
    uint32_t offset;
    struct rte_mbuf* next;
    struct rte_mbuf* prev;
};
```

Atributy `seq_number` a `segment_size` udávají sekvenční číslo daného TCP segmentu a jeho velikost bez TCP hlavičky. Uložení těchto atributů slouží k tomu, aby aplikace nemusela při každém nahlédnutí do paketu v bufferu znovu zpracovávat jeho L2-L4 hlavičky k získání těchto hodnot. Atribut `offset` udává, kde v datové části segmentu má buffer začít hledat další DNS zprávu. Tato hodnota je užitečná, pokud z daného paketu byla již vyextrahována alespoň jedna DNS zpráva, ale paket ještě obsahuje další DNS zprávu, která se do něj již celá nevešla. Její zbytek by tedy měl být odeslán v dalším paketu v pořadí a první paket tedy musí být zařazen do bufferu a počkat na zbytek této DNS zprávy.

Atributy `next` a `prev` jsou ukazatele na následující a předchozí položky v seznamu. Jejich datové typy ovšem nejsou `struct TcpSegment`, ale `struct rte_mbuf`. `rte_mbuf` je datová struktura, ve které framework DPDK představený v kapitole 4, uchovává všechny příchozí pakety. Aby se nemusela pro položky seznamu alokovat dodatečná paměť, jsou struktury `TcpSegment` vnořeny do struktur `rte_mbuf`. Princip tohoto uspořádání je zobrazen na obrázku 10.2.



Obrázek 10.2: Vnoření TcpSegment struktury do rte_mbuf

Struktura `rte_mbuf` kromě samotného paketu obsahuje také některé informace o daném paketu. Mezi tyto informace patří například délka paketu, hodnota RSS hashe pro daný paket, verze IP protokolu aj. Za těmito informacemi poté následuje samotný paket. Nenásleduje ovšem bezprostředně za těmito informacemi, ale s určitým odstupem přizpůsobeným pro zarovnaný přístup do paměti cache. Prostor mezi koncem informací o paketu a samotným paketem se nazývá *headroom*. Tento prostor jde nastavit tak, aby měl určitou minimální velikost a uživatel měl možnost do něj uložit svoje doplňující data. Právě do tohoto prostoru aplikace ukládá strukturu `TcpSegment`, čímž šetří pamětí a dodatečnými voláními pro její alokaci. Pokud se daný paket nevejde do jedné struktury `rte_mbuf`, je rozdělen mezi více těchto struktur, které se na sebe vzájemně odkazují.

Buffer řadí jednotlivé pakety podle jejich sekvenčních čísel. Sekvenční čísla nově příchozích paketů jsou porovnávána s ISN pro danou stranu spojení a s následujícím očekávaným sekvenčním číslem. Pokud sekvenční číslo paketu odpovídá následujícímu očekávanému sekvenčnímu číslu, je paket okamžitě zpracován. Pokud jsou z něj vyextrahovány všechny DNS zprávy, jsou tyto zprávy poslány na další zpracování modulu *Transakční tabulka* a paket je přeposlán dále. Pokud ale paket obsahuje neúplnou DNS zprávu, musí být zařazen do bufferu, kde počká na doručení paketu se zbytkem této DNS zprávy.

Pokud sekvenční číslo paketu neodpovídá následujícímu očekávanému sekvenčnímu číslu, musí aplikace rozhodnout, jestli je to paket doručený mimo pořadí, nebo opětovně zaslaný paket. Pokud je sekvenční číslo paketu menší než následující očekávané sekvenční číslo, tento paket již byl zpracován a aplikace jej tedy znovu nezpracovává. Pokud je jeho sekvenční číslo větší než následující očekávané sekvenční číslo, došel tento paket mimo pořadí a je tedy vložen na odpovídající pozici do bufferu. Aplikace si při těchto porovnáváních musí dát pozor na přetečení sekvenčního čísla. Sekvenční čísla zabírají v TCP hlavičce 4 byty a mají tedy odpovídající rozsah od 0 do $2^{32}-1$. Pokud si spojení zvolí ISN blízko maximální hodnotě tohoto rozsahu a přenese během svého průběhu větší objem dat, může se stát, že sekvenční čísla paketů tohoto spojení přetečou a začnou opět od nuly. Tento stav představuje potíž, pokud má následující očekávané sekvenční číslo hodnotu těsně u maximálního rozsahu a přijde paket mimo pořadí, jehož sekvenční číslo již přeteklo. V tomto případě je nutné sekvenční číslo porovnat ještě s ISN dané strany spojení. Jestliže je sekvenční číslo paketu menší než následující očekávané sekvenční číslo a zároveň i menší než ISN, je paket pouze doručen mimo pořadí a má být zařazen na příslušné místo do bufferu. Jestliže je sekvenční číslo takového paketu větší než ISN, jedná se o opětovné zaslání již zpracovaného paketu a tento paket tedy aplikace nemusí znovu zpracovávat.

Kapitola 11

Testování zpracování DNS přes TCP

Testování zpracování DNS přes TCP opět probíhalo ve dvou fázích – testy korektnosti a výkonnostní testy pro ověření dopadu na výkon aplikace. Testy korektnosti spočívaly ve spuštění aplikace DPDK DNS sonda nad vzorovými PCAP soubory s různými scénáři DNS provozu přes protokol TCP.

Testovány byly jednoduché TCP toky, ve kterých pakety došly ve správném pořadí a ve kterých v každém datovém paketu byla jedna celá DNS zpráva. V takovém případě aplikace nemusela vůbec využít přeuspořádacího bufferu a testována tedy byla hlavně implementace konečného automatu pro udržování stavu TCP spojení. Dále byly testovány případy, kdy odeslané DNS zprávy byly příliš velké pro jeden paket a byly tedy rozděleny mezi více segmentů. Zde již aplikace musela pro rekonstrukci datového toku využít přeuspořádací buffer. Přeuspořádací buffer musel být využit i v případech, kdy pakety TCP toku byly odeslány mimo pořadí. Výsledky testů korektnosti ukázaly očekávané chování aplikace nad všemi testovanými scénáři.

Výkonnostní testy probíhaly stejně jako při předchozích měřeních v kapitolách 6 a 8 v síťové laboratoři L311 na Fakultě informačních technologií Vysokého učení technického v Brně. Síťový provoz DNS paketů přes protokol TCP byl na 10 Gbps lince simulován pomocí generátoru síťového provozu Spirent TestCenter SPT-2000A. Aplikace DPDK DNS sonda byla spuštěna na serveru s nainstalovaným frameworkem DPDK verze 18.05 s následující hardwarovou konfigurací:

- **CPU:** Intel Xeon E5-2620@2.00 GHz
- **RAM:** 32 GB DDR3@1600 MHz
- **Operační systém:** Debian 8.0 "Jessie", jádro Linux 4.9.28-2
- **NIC:** Intel X-520-2, 2x 10 Gb

11.1 DNS přes TCP bez přeuspořádacího bufferu

Jako první scénář pro výkonnostní testy byl zvolen jednoduchý TCP tok, ve kterém byly všechny pakety ve správném pořadí a ve kterém každý datový paket obsahoval vždy přesně jednu celou DNS zprávu. Tímto scénářem byla testována zátěž aplikace při zpracování DNS přes TCP bez použití přeuspořádacího bufferu. Řazení paketů uvnitř přeuspořádacího

bufferu totiž obsahuje výpočetně náročné operace, u kterých byl odhadován citelný dopad na celkový výkon aplikace. Proto byl nejdříve měřen výkon aplikace bez použití tohoto bufferu, aby byly ustanoveny základní hodnoty pro nasazení aplikace v ideálním provozu, zvláště když přenos DNS přes TCP tvoří minimum celkového reálného DNS provozu [5]. Výsledky tohoto měření jsou zobrazeny v tabulce 11.1.

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing s EDNS	5,5	9,8	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka	5,4	9,7	10	10	10	10	10	10	10	10
DNS parsing s EDNS + Transakční tabulka + Parquet export	2,6	5	7	9	9,5	9,5	9,5	9,7	9,7	9,7

Tabulka 11.1: Zpracování DNS přes TCP bez přeuspořádacího bufferu na 10 Gbps lince

Aplikace byla opět postupně spouštěna jen s některými částmi zpracování paketu. Měření bylo prováděno vždy aplikací s plným parsováním DNS včetně vyhledávání EDNS záznamu, protože zde již nebyl předmětem měření výkon samotného zpracování DNS zprávy jako v kapitole 8. Pro ustanovení základních hodnot byla nejdříve změřena aplikace v režimu prostého přeposílání paketů bez jakéhokoliv zpracování. Zde již jedno pracovní jádro aplikace dokáže zpracovat celou 10 Gbps linku.

Dále byla aplikace spuštěna pouze s modulem *Parsování paketu*, který provede zpracování L2-L4 hlaviček a zpracování samotné DNS zprávy. V tomto případě již tři pracovní jádra dokáží obsloužit celou 10 Gbps linku. Při přidání modulu *Transakční tabulka*, který provádí párování dvojic dotaz-odpověď je výkon aplikace skoro totožný. To je způsobeno tím, že TCP provoz je z nezanedbatelné části tvořen pakety řídícími dané spojení, které neobsahují DNS zprávy. Oproti UDP tedy při stejném počtu přenesených paketů obsahuje TCP provoz menší počet DNS zpráv a modul *Transakční tabulka* tedy není tolik zatížen. Stejný princip způsobí, že při přidání exportu DNS záznamů do Parquet formátu dosáhne aplikace vyššího výkonu než při zpracování UDP provozu zobrazeném v tabulce 8.4 v sekci 8.2. Aplikace v tomto případě dokáže zpracovat až 2,6 Gbps TCP provozu na jednom pracovním jádře.

11.2 DNS přes TCP s přeuspořádacím bufferem

Jako další výkonnostní test byl zvolen složitější TCP tok, který již obsahoval velké DNS zprávy rozprostřené přes více paketů. Pro tento tok již aplikace musí využít řazení paketů v přeuspořádacím bufferu. Výsledky měření jsou zobrazeny v tabulce 11.2.

Typ měření (v Gbps) / počet worker jader	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
L2-FWD	10	10	10	10	10	10	10	10	10	10
DNS parsing s EDNS	0,9	0,9	0,9	0,8	0,8	0,7	0,7	0,7	0,7	0,7
DNS parsing s EDNS + Transakční tabulka	0,9	0,9	0,8	0,8	0,7	0,7	0,7	0,7	0,7	0,7
DNS parsing s EDNS + Transakční tabulka + Parquet export	0,9	0,8	0,8	0,7	0,6	0,6	0,6	0,6	0,5	0,5

Tabulka 11.2: Zpracování DNS přes TCP s přeuspořádacím bufferem na 10 Gbps lince

Bez ohledu na to, se kterými moduly byla aplikace DPDK DNS sonda spuštěna, se ukázala jako maximum pro zpracování DNS přes TCP hodnota 0,9 Gbps. Této hodnoty bylo dosaženo již při spuštění aplikace pouze s modulem *Parsování paketu*. To potvrzuje

odhady, že omezujícím faktorem při výkonu aplikace je právě přeuspořádací buffer. Při spuštění aplikace s více pracovními jádry již nedochází ke škálování výkonu, ale aplikace si udržuje stále stejnou či dokonce lehce klesající výkonnost.

Důvodem k tomuto omezení je to, že pakety v přeuspořádacím bufferu jsou uloženy ve strukturách `rte_mbuf`. Tyto struktury framework DPDK alokuje v kontinuálních paměťových blocích (tzv. mempoolech) ve velkých paměťových stránkách po spuštění aplikace. I když je možné vytvořit několik těchto paměťových bloků, aplikace DPDK DNS sonda momentálně využívá pro všechny příchozí pakety pouze jeden velký paměťový blok, ze kterého získává struktury `rte_mbuf`. Paket většinou obsazuje strukturu `rte_mbuf` od momentu přijetí paketu do momentu odeslání paketu dále po síti, kdy je příslušná struktura `rte_mbuf` uvolněna pro použití dalším paketem. Při vložení paketu do přeuspořádacího bufferu je ale příslušná struktura `rte_mbuf` obsazena paketem i po jeho odeslání dále po síti a to až do chvíle, kdy je zpětně zrekonstruován daný TCP tok. Důsledkem toho je, že pokud aplikace drží v přeuspořádacích bufferech všech aktivních TCP spojení příliš mnoho paketů, nezbývá v paměťovém bloku dostatečné množství volných struktur `rte_mbuf` pro další příchozí pakety a snižuje se celkový výkon aplikace.

Protože každé pracovní jádro aplikace má svoji vlastní tabulku aktivních TCP spojení, větší počet pracovních jader může způsobovat větší a rychlejší zaplnění paměťového bloku pro struktury `rte_mbuf`, což způsobuje o něco nižší výkon aplikace při spuštění s více pracovními jádry. Hashovací tabulka v aplikaci DPDK DNS sonda, která je použita pro tabulku aktivních TCP spojení, implementuje mechanismus pro průběžné mazání neaktivních záznamů. Ani tento mechanismus ovšem nestíhá efektivně mazat záznamy z tabulky aktivních TCP spojení a tím uvolňovat struktury `rte_mbuf`. Možná řešení tohoto problému jsou nastíněna v následující kapitole.

Kapitola 12

Další vývoj aplikace DPDK DNS sonda

Tato diplomová práce měla za cíl vhodně rozšířit aplikaci DPDK DNS sonda pro zpracování DNS provozu a export jeho statistik. V předchozích kapitolách byla představena dvě implementovaná rozšíření – optimalizované parsování DNS zpráv a zpracování provozu DNS přes transportní protokol TCP. Vývoj aplikace DPDK DNS sonda však v době psaní této práce stále pokračuje a nabízí se hned několik vhodných oblastí k vylepšení stávajícího stavu aplikace:

- **Přesun parsování hlaviček paketů do HW**

  Optimalizace parsování DNS zpráv představená v této práci dosáhla nad očekávání dobrých výsledků. Bylo dosaženo až několikanásobného zrychlení oproti knihovně *libknot*, kterou aplikace používala dosud. Největšího zrychlení bylo dosaženo při zpracování DNS paketů obsahujících EDNS záznamy. Původním záměrem při vývoji aplikace bylo ji nasadit na specializovanou síťovou kartu, která by byla připojena v síti před zařízením s běžícím DNS serverem. Tato síťová karta by měla obsahovat programovatelné hradlové pole FPGA. Tohoto pole by bylo možné využít k další optimalizaci parsování hlaviček paketu a případně i těla DNS zprávy. Přesun této části aplikace ze softwaru na programovatelné hradlové pole by při správné implementaci ještě více urychlil zpracování paketů a dovolil by aplikaci více se soustředit na párování dvojic dotaz-odpověď a na export statistik provozu.

- **Optimalizace zpracování DNS přes TCP**

  Tato práce představila funkční řešení pro zpracování DNS provozu využívajícího transportní protokol TCP. Výkonnostní testy ale odhalily, že toto řešení značně snižuje výkon aplikace. Důvodem je způsob, jakým aplikace DPDK DNS sonda alokuje paměť pro zpracovávané pakety. Pakety jsou ukládány v strukturách `rte_mbuf`, které jsou brány z jednoho kontinuálního paměťového bloku společného pro všechna pracovní jádra aplikace. Pokud se tento paměťový blok zaplní, jsou ovlivněna všechna pracovní jádra aplikace.

  Jedním řešením by mohla být alokace kontinuálního paměťového bloku pro struktury `rte_mbuf` pro každé pracovní jádro zvlášť. Toto řešení by mohlo dovolit aplikaci opět škálovat výkon podle počtu spuštěných pracovních jader. Pokud by některému pracovnímu jádru došlo místo v paměťovém bloku, byl by ovlivněn výkon pouze tohoto

jednoho pracovního jádra. Problémem tohoto řešení je určení velikosti paměťového bloku pro každé pracovní jádro. Výhodou jednoho paměťového bloku pro všechna paměťová jádra dohromady je to, že je možné velikost tohoto bloku poměrně spolehlivě nadimenzovat pro zvládnutí zpracování celé 10 Gbps linky. Pokud by mělo každé pracovní jádro svůj vlastní paměťový blok, bylo by těžké určit jeho optimální velikost. Velikost nutná pro zpracování 10 Gbps linky by se mohla rovnoměrně rozdělit podle počtu pracovních jader. Tato varianta by ale selhala, pokud by provoz nebyl rovnoměrně rozprostřen mezi všechna pracovní jádra. Přidělování paketů jednotlivým pracovním jádrům probíhá v aplikaci podle identifikace síťového toku pěticí – zdrojová a cílová IP adresa, zdrojový a cílový port, transportní protokol. V případě vytížení větší části 10 Gbps linky jedním tokem by tento provoz zpracovávalo pouze jedno pracovní jádra, kterému by poté nestačil jeho paměťový blok pro struktury `rte_mbuf`. Pokud by každé pracovní jádro dostalo přiděleno paměťový blok, který by byl dostatečně velký pro zvládnutí celé 10 Gbps linky, narostla by celková paměťová náročnost aplikace na neúnosnou mez.

Dalším možným řešením by bylo vytvořit kontinuální paměťový blok pouze pro účely ukládání paketů v přeuspořádacím bufferu. Tímto způsobem by pakety uložené v přeuspořádacím bufferu neovlivňovaly dostupnost struktur `rte_mbuf` pro nově příchozí pakety a neovlivňovaly tak negativně výkon aplikace. Nevýhodou tohoto řešení je opět zvýšení paměťové náročnosti aplikace o tento nový paměťový blok. Další nevýhodou ovlivňující výkon aplikace je také nutnost překopírovat obsah paketu při vložení do přeuspořádacího bufferu. Paket by musel být překopírován ze struktury `rte_mbuf` alokované v paměťovém bloku pro příchozí pakety do struktury `rte_mbuf` alokované v paměťovém bloku pro přeuspořádací buffer.

- **Optimalizace exportu statistik do Parquet formátu**

  Měření v sekci 6.1 ukázala, že velkým faktorem ovlivňujícím výkon aplikace je export statistik do formátu Parquet. Export do tohoto formátu také negativně ovlivňuje paměťovou náročnost aplikace. Jelikož se jedná o sloupcový datový formát, je nutné před zápisem na disk nejdříve akumulovat dostatečné množství dat v paměti aplikace, aby byla data v tomto formátu efektivně uložena.

  Možným řešením tohoto problému by bylo vytvoření samostatné aplikace pro zpracování formátu Parquet. Aplikace DPDK DNS sonda by ukládala statistiky provozu na disk v surové podobě ve svém interním formátu datové struktury *DnsRecord*. Samostatná aplikace by poté četla soubory s těmito daty a konvertovala je do formátu Parquet.

Kapitola 13

Závěr

Cílem této diplomové práce bylo nastudovat protokol DNS a seznámit se s aplikací DPDK DNS sonda, která provádí detailní monitorování DNS provozu ve vysokorychlostních sítích. Dalším úkolem bylo identifikovat slabiny aplikace a navrhnout a implementovat možná rozšíření, která by je eliminovala. V úvodní kapitole byla představena architektura systému DNS a jeho jednotlivé komponenty. Bylo popsáno jak fungování jednotlivých komponent, tak jejich propojení tvořící celosvětovou službu DNS.

Aplikace DPDK DNS sonda je implementována v programovacím jazyku C++ za pomoci frameworku DPDK. Tento framework poskytuje knihovny a ovladače pro zpracování paketů ve vysokorychlostních sítích. K určení částí aplikace, které nejvíce zpomalují provoz, bylo provedeno důkladné testování pomocí generátoru síťového provozu Spirent TestCenter. Pro parsování paketů je použita knihovna *libknot*, která se ukázala jako hlavní limitující faktor při zpracování paketů. Bylo navrženo a implementováno rozšíření aplikace, které upustí od použití knihovny *libknot* a místo toho provede vlastní parsování DNS hlavičky a dat paketu. Hlavní zrychlení při tomto zpracování paketu přichází díky absenci dynamické alokace paměti během parsování DNS dat. Výkonnostní testy implementovaného rozšíření ukázaly až několikanásobné zrychlení oproti knihovně *libknot*.

Další fáze diplomové práce se zabývala rozšířením aplikace DPDK DNS sonda o podporu zpracování DNS provozu přes transportní protokol TCP. V práci jsou popsány základní principy protokolu TCP, zejména ty vztahující se ke zpracování DNS provozu. Bylo navrženo a implementováno rozšíření aplikace, které umožňuje spolehlivou rekonstrukci DNS dat přenášených přes TCP spojení. Testování odhalilo očekávaný dopad na výkon aplikace při zpracování tohoto typu provozu. V práci jsou popsána možná řešení pro limitování tohoto dopadu na výkon. Na závěr jsou nastíněny možnosti dalšího vývoje aplikace DPDK DNS sonda.

Literatura

[1] Benvenuti, C.: *Understanding Linux network internals*. Sebastapol, CA: O'Reilly, c2006, ISBN 978-0-12-088588-6.

[2] Cheng, Y.; Chu, J.; Radhakrishnan, S.; aj.: TCP Fast Open. RFC 7413, RFC Editor, December 2014, [Online; navštíveno 18.05.2019].
URL https://www.rfc-editor.org/rfc/rfc7413.txt

[3] Contavalli, C.; van der Gaast, W.; Lawrence, D.; aj.: Client Subnet in DNS Queries. RFC 7871, RFC Editor, May 2016, [Online; navštíveno 15.01.2019].
URL https://www.rfc-editor.org/rfc/rfc7871.txt

[4] Crocker, S.; Rose, S.: Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC). RFC 6975, RFC Editor, July 2013, [Online; navštíveno 15.01.2019].
URL https://www.rfc-editor.org/rfc/rfc6975.txt

[5] CZ.NIC, z. s. p. o.: *Dotazy podle IP protokolu - Statistiky CZ*. 2019, [Online; navštíveno 18.05.2019].
URL https://stats.nic.cz/stats/dns_queries_by_protocol/

[6] CZ.NIC, z. s. p. o.: *Průměrný počet dotazů za vteřinu - Statistiky CZ*. 2019, [Online; navštíveno 15.01.2019].
URL https://stats.nic.cz/stats/dns_queries_per_second/

[7] Damas, J.; Graff, M.; Vixie, P.: Extension Mechanisms for DNS (EDNS(0)). STD 75, RFC Editor, April 2013, [Online; navštíveno 15.01.2019].
URL https://www.rfc-editor.org/rfc/rfc6891.txt

[8] Dickinson, J.; Dickinson, S.; Bellis, R.; aj.: DNS Transport over TCP - Implementation Requirements. RFC 7766, RFC Editor, March 2016, [Online; navštíveno 14.05.2019].
URL https://www.rfc-editor.org/rfc/rfc7766.txt

[9] Doležal, P.: *Řízení provozu ve vysokorychlostních sítích v prostředí DPDK*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2017.
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=19786

[10] DPDK: *2. Overview - Data Plane Development Kit 19.02.0-rc1 documentation*. 2019, [Online; navštíveno 02.01.2019].
URL http://doc.dpdk.org/guides/prog_guide/overview.html

[11] DPDK: *2. System Requirements - Data Plane Development Kit 19.02.0-rc2 documentation*. 2019, [Online; navštíveno 16.01.2019].
URL http://doc.dpdk.org/guides/linux_gsg/sys_reqs.html#use-of-hugepages-in-the-linux-environment

[12] DPDK: *3. Environment Abstraction Layer - Data Plane Development Kit 19.02.0-rc1 documentation*. 2019, [Online; navštíveno 02.01.2019].
URL http://doc.dpdk.org/guides/prog_guide/env_abstraction_layer.html

[13] DPDK: *5. Linux Drivers - Data Plane Development Kit 19.05.0 documentation*. 2019, [Online; navštíveno 14.05.2019].
URL https://doc.dpdk.org/guides/linux_gsg/linux_drivers.html

[14] DPDK: *7. Poll Mode Driver - Data Plane Development Kit 19.02.0-rc1 documentation*. 2019, [Online; navštíveno 02.01.2019].
URL http://doc.dpdk.org/guides/prog_guide/poll_mode_drv.html

[15] DPDK: *Supported hardware*. 2019, [Online; navštíveno 02.01.2019].
URL https://core.dpdk.org/supported/

[16] Eastlake, D. E.: Domain Name System Security Extensions. RFC 2535, RFC Editor, March 1999, [Online; navštíveno 14.01.2019].
URL http://www.rfc-editor.org/rfc/rfc2535.txt

[17] Everhart, C.; Mamakos, L.; Ullmann, R.; aj.: New DNS RR Definitions. RFC 1183, RFC Editor, October 1990, [Online; navštíveno 14.01.2019].
URL https://www.rfc-editor.org/rfc/rfc1183.txt

[18] Foundation, A. S.: *Apache Parquet*. 2018, [Online; navštíveno 15.01.2019].
URL https://parquet.apache.org/documentation/latest/

[19] Gustafsson, A.: Handling of Unknown DNS Resource Record (RR) Types. RFC 3597, RFC Editor, September 2003, [Online; navštíveno 14.01.2019].
URL https://www.rfc-editor.org/rfc/rfc3597.txt

[20] Herbert, T.; de Bruijn, W.: *Scaling in the Linux Networking Stack*. 2017, [Online; navštíveno 06.01.2019].
URL https://www.kernel.org/doc/Documentation/networking/scaling.txt

[21] Lewis, E.; Hoenes, A.: DNS Zone Transfer Protocol (AXFR). RFC 5936, RFC Editor, June 2010, [Online; navštíveno 18.05.2019].
URL https://www.rfc-editor.org/rfc/rfc5936.txt

[22] Matoušek, P.: *Síťové aplikace a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014, ISBN 978-80-214-3766-1.

[23] Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS. RFC 3401, RFC Editor, October 2002, [Online; navštíveno 14.01.2019].
URL https://www.rfc-editor.org/rfc/rfc3401.txt

[24] Melnik, S.; Gubarev, A.; Long, J. J.; aj.: Dremel: Interactive Analysis of Web-Scale Datasets. In *Proc. of the 36th Int'l Conf on Very Large Data Bases*, 2010, s. 330–339.

[25] Mockapetris, P.: Domain names - concepts and facilities. STD 13, RFC Editor, November 1987, [Online; navštíveno 14.01.2019].
URL http://www.rfc-editor.org/rfc/rfc1034.txt

[26] Mockapetris, P.: Domain names - implementation and specification. STD 13, RFC Editor, November 1987, [Online; navštíveno 14.01.2019].
URL http://www.rfc-editor.org/rfc/rfc1035.txt

[27] Piat, F.; Capper, S.; aj.: *Hugepages - Debian Wiki*. Červen 2017, [Online; navštíveno 02.01.2019].
URL https://wiki.debian.org/Hugepages

[28] Postel, J.: Transmission Control Protocol. STD 7, RFC Editor, September 1981, [Online; navštíveno 14.05.2019].
URL http://www.rfc-editor.org/rfc/rfc793.txt

[29] Postel, J.; Reynolds, J.: File Transfer Protocol. STD 9, RFC Editor, October 1985, [Online; navštíveno 20.05.2019].
URL http://www.rfc-editor.org/rfc/rfc959.txt

[30] Seth, S.; Venkatesulu, M. A.: *TCP/IP architecture, design and implementation in Linux*. Hoboken, N.J. Los Alamitos, CA: Wiley IEEE Computer Society, 2008, ISBN 978-0470-14773-3.

[31] Wellington, B.; Gudmundsson, O.: Redefinition of DNS Authenticated Data (AD) bit. RFC 3655, RFC Editor, November 2003, [Online; navštíveno 14.01.2019].
URL https://www.rfc-editor.org/rfc/rfc3655.txt

[32] Woo, S.; Park, K.: *Scalable TCP Session Monitoring with Symmetric Receive-side Scaling*. KAIST Technical Report, 2012, [Online; navštíveno 06.01.2019].
URL http://www.ndsl.kaist.edu/~kyoungsoo/papers/TR-symRSS.pdf

Příloha A

Obsah přiloženého paměťového média

- **src** – adresář zdrojových souborů aplikace DPDK DNS sonda
- **xdolez64** – adresář obsahující technickou zprávu ve formátu pdf
- **latex** – adresář obsahující technickou zprávu ve formátu tex
- **pcaps** – adresář s některými vzorovými PCAP soubory, na kterých byla aplikace testována
- **README** – soubor s pokyny k instalaci a spuštění aplikace

Příloha B

Statistiky exportované aplikací DPDK DNS sonda

Název položky	Datový typ	Význam položky
ID	Int32	ID v DNS hlavičce
UnixTime	Int64	Sekundy od začátku unixové Epochy
Time	Int64	Čas od začátku unixové Epochy
Qname	String	Doménové jméno v dotazu
Domainname	String	Poslední dvě domény doménového jména v dotazu
Len	Int32	Velikost paketu s dotazem
Frag	Int32	Fragmentace
TTL	Int32	Time To Live z IP hlavičky
IPv	Int32	Verze IP protokolu
Prot	Int32	Transportní protokol
Src	String	Zdrojová (klientská) IP adresa
SrcPort	Int32	Zdrojový (klientský) port
Dst	String	Cílová (serverová) IP adresa
DstPort	Int32	Cílový (serverový) port
UDPSum	Int32	Checksum v UDP hlavičce
DNSLen	Int32	Velikost DNS části paketu s dotazem
ResLen	Int32	Velikost paketu s odpovědí
TimeMicro	Int64	Mikrosekundy od začátku unixové Epochy
DNSResLen	Int32	Velikost DNS části paketu s odpovědí
AA	Boolean	AA bit v DNS hlavičce odpovědi
TC	Boolean	TC bit v DNS hlavičce odpovědi
RD	Boolean	RD bit v DNS hlavičce dotazu
RA	Boolean	RA bit v DNS hlavičce odpovědi
Z	Boolean	Z bit v DNS hlavičce dotazu
AD	Boolean	AD bit v DNS hlavičce odpovědi
CD	Boolean	CD bit v DNS hlavičce dotazu
AnCount	Int32	Počet záznamů v sekci s odpověďmi
ArCount	Int32	Počet záznamů v sekci s doplňujícími informacemi
NsCount	Int32	Počet záznamů v sekci s autoritativními servery
QdCount	Int32	Počet záznamů v sekci s dotazy

OpCode	Int32	Operační kód v DNS hlavičce dotazu
RCode	Int32	Chybový kód v DNS hlavičce odpovědi
QType	Int32	Typ DNS dotazu
QClass	Int32	Třída DNS dotazu
EdnsUDP	Int32	Maximální velikost UDP podle EDNS záznamu
EdnsVersion	Int32	Verze EDNS záznamu
EdnsDO	Boolean	DO bit v EDNS záznamu
EdnsNSID	String	Obsah NSID položky v EDNS záznamu
EdnsDnssecDau	String	Seznam povolených DNSSEC DAU algoritmů
EdnsDnssecDhu	String	Seznam povolených DNSSEC DHU algoritmů
EdnsDnssecN3u	String	Seznam povolených DNSSEC N3U algoritmů

Tabulka B.1: Statistiky exportované aplikací DPDK DNS sonda