VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

AKCELEROVANÝ FIREWALL S DPDK

DPDK ACCELERATED FIREWALL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ HOLUBÁŘ**
AUTHOR

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

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Holubář Jiří**
Obor: Informační technologie
Téma: **Akcelerovaný firewall s DPDK**
DPDK Accelerated Firewall
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s frameworkem DPDK.
2. Nastudujte problematiku síťových firewallů.
3. Navrhněte firewall nad frameworkem DPDK s podporou vícevláknového zpracování.
4. Implementujte navržený firewall.
5. Otestujte funkčnost implementované aplikace
6. Změřte propustnost implementované aplikace.
7. Zhodnoťte dosažené výsledky a diskutujte další možnosti práce.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Vrána Roman, Ing.**, UPSY FIT VUT
Konzultant: Dražil Jan, Ing., UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

V dnešní době, kdy téměř každý využívá Internet, je třeba zajistit také zabezpečení provozu na sítích. K tomuto napomáhá firewall. Na některých linkách je ovšem vyžadována vyšší propustnost než na jiných. Tato práce zkoumá možnosti využití knihovny DPDK při implementaci firewallu za účelem dosažení co největší propustnosti.

Abstract

Nowadays, when almost everyone uses the Internet, network traffic security must also be ensured. This is what firewall helps with. Some routes require higher bandwidth than others. This thesis explores possibilities of using the DPDK library when implementing the firewall in order to achieve the highest possible bandwidth.

Klíčová slova

firewall, DPDK, ICMP, klasifikace paketů, TCP, ACL, UDP, IPv4, firewall ve FreeBSD

Keywords

firewall, DPDK, ICMP, packet classification, TCP, ACL, UDP, IPv4, firewall in FreeBSD

Citace

HOLUBÁŘ, Jiří. *Akcelerovaný firewall s DPDK*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Roman Vrána,

Akcelerovaný firewall s DPDK

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Romana Vrány. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Holubář
17. května 2017

Poděkování

Na tomto místě bych rád poděkoval Ing. Romanu Vránovi za ochotu a čas, který mi při vedení mé práce věnoval.

Obsah

Kapitola 1

Úvod

V dnešní době Internet spojuje téměř celý svět. Toto propojení přináší spousty výhod, také však přináši mnoho bezpečnostních rizik. Některým se mohou uživatelé vyhnout sami, například pomocí silných hesel u svých účtů nebo opatrností při otevírání příloh e-mailů. Některým rizikům se ovšem uživatel pouhou opatrností vyhnout nemůže. Proto má většina systémů v sobě dnes již zabudovaný firewall, který dokáže uživatele ochránit snížit tak rizika využívání Internetu.

Firewall ovšem neslouží pouze běžným uživatelům, ale například i poskytovatelům Internetu. Ti díky němu mohou lépe kontrolovat provoz na síti. Mohou například zakázat přeposílání zpráv na určité porty, které využívají nelegální aplikace.

Cílem této práce je popsat, jak fungují různé druhy firewallu. Dále pak jaké jsou jeho dostupné implementace v různých OS. Následně jsou popsány protokoly, které se využívají při přenosu paketů po síti. V další části je popsán návrh implementace firewallu s využitím knihovny DPDK. V následujících kapitolách je popsána knihovna DPDK a dále implementace firewallu pomocí této knihovny.

Kapitola 2

Firewall

Firewall je bezpečnostní systém, který má za úkol kontrolovat a filtrovat provoz na síti podle předem definovaných pravidel. Má za úkol dohlížet na provoz paketů a rozhodovat, co se s nimi stane. Pokud paket nevyhovuje žádnému z definovaných pravidel je následně zahozen. Může být provozován zároveň s nástrojem pro překlad síťových adres, který umožňuje zařízením v síti, které mají neveřejné IP adresy, komunikovat se zařízeními na internetu pomocí veřejné IP adresy nebo určeného rozsahu IP adres.

V dnešní době se firewally používají na většině zařízeních v síti. Obsahují je operační systémy a i většina routerů v sítích obsahuje nějaký firewall.

Nutnost využívat firewall vznikla, když začaly společnosti využívat kromě své interní sítě i internet. Před příchodem firewallu totiž zajišťoval bezpečnost pouze ACL (access control list), který využívaly routery [18]. Ten dokázal určit které IP adresy budou mít povolený nebo zamítnutý přístup do sítě. Toto bylo ovšem nedostatečné zabezpečení. Proto v roce 1992 firma Digital Equipment Cortp vyvinula první komerční firewall nazývaný DEC SEAL.

2.1 Paketové filtry

První firewally fungovaly jako paketové filtry. Každému paketu, který chtěl projít přes tento firewall byla zkontrolována zdrojová a cílová IP adresa, číslo portu a protokol. Tyto parametry byly vyhodnoceny pravidly, a pokud některý z nich nevyhovoval, byl paket zahozen.

Tyto filtry fungovaly převážně na dvou vrstvách OSI modelu (L2 a L3). Každý paket vyhodnocovaly samostatně, tudíž nebyly schopné určit, zda pakety patří k nějakému toku či ne. Díky tomu nebyly schopné vyhodnocovat složitější pravidla.

2.2 Stavový firewall

Kvůli nutnosti rozeznat stavy jednotlivých připojení, začaly firewally zaznamenávat stav jednotlivých spojení, aby poznaly, které pakety začínají nové spojení, které patří k již existujícímu spojení, a které nepatří k žádnému. Díky tomu byla mohl být udělen nebo zamítnut přístup paketům podle historie stavové tabulky.

2.3 Firewall aplikační vrstvy

Další částí vývoje firewallů byly ty, které dokázaly ochránit aplikace běžící na stroji. Tyto firewally umožňovaly filtrovat pakety na všech vrstvách OSI modelu až po aplikační. Výhodou tohoto firewallu je, že dokáže blokovat komunikaci na aplikační úrovni. může například zablokovat malware nebo určité nežádoucí webové stránky.

Tyto firewally využívají DPI (deep packet inspection), což je zkoumání dat jednotlivých paketů. O přeposlání paketu se tedy nerozhoduje pouze na základě jeho hlavičky, ale i dat obsažených v něm. Díky DPI je například možné přidělovat zdroje určitým tokům [5]. Například zprávy označené vyšší prioritou budou přeposlány dříve, než zprávy s nižší prioritou.

2.4 Proxy firewall

Tento firewall pracuje také na aplikační vrstvě. Funguje jako zprostředkovatel pro dotazy z jedné sítě do druhé pro specifické síťové aplikace. Zabraňuje přímé komunikaci mezi oběma stranami spojení, tudíž jsou nuceny komunikovat pouze přes proxy, který může komunikaci filtrovat podle předem určených pravidel [6]. Služba proxy musí být ale spuštěna zvlášť pro každou internetovou aplikaci, kterou firewall podporuje.

Kapitola 3

Dostupné implementace firewallu

3.1 Firewall ve Windows

Ochranu proti hrozbám při využívání internetu poskytují v systémech Windows služby Windows Firewall a Windows Defender [17].

Zatímco Windows Defender je spíše antivirus, který chrání před malwarem anežádoucím obsahem, nástoj Windows Firewall je klasický stavový firewall, který je možné nastavovat v nastavení systému Windows. OS umožňuje nastavení firewallu na tři základní profily. Tyto profily se mění v závislosti na síti, ke které je počítač připojen. Tyto profily se nazývají Veřejný, Soukromý a Doménový. Veřejný profil má nejvíce omezení, neboť je využíván na veřejných sítích, kde hrozí největší nebezpečí nežádoucího provozu. Soukromý profil je využíván na soukromých sítích, ve kterých jsou bezpečnostní rizika menší. Doménový profil je nejméně omezující, neboť předpokládá. že síť, do které je připojen, je dobře chráněna. Tento profil je nastaven po připojení počítače k síti s doménou, která je pro počítač duvěryhodná.

V nastavení je samozřejmě možné také ručně nastavit firewallu pravidla. Je možné povolit nebo zakázat určitě porty, protokoly, adresy nebo rovnou zamezit odesílání a přijímání dat některých programů.

3.2 Firewall v Linux systémech

Systémy Linux používají jako firewall NETFILTER. Jeho zjednodušené chování je na obrázku 3.1. Tento nástroj umožňuje nejen filtrování provozu, ale také například překlad síťových adres. Pro funkce firewallu zde slouží iptables. Jak už název napovídá, iptables jsou v podstatě tabulky adres, portů a pravidel pro jejich další zpracování.

Tato pravidla jsou seskupeny do řetězců [10]. U příchozího paketu je na základě jeho původu rozhodnuto, který řetězec začne procházet. Poté se paket porovnává s jednotlivými pravidli v řetězci. V případě, že paket nevyhovuje pravidlu, pokračuje na další pravidlo. Pokud pravidlu vyhovuje, provede se akce, která je u něj nastavená. Pravidlo může paket například odeslat paket dojiného řetězce, vrátit porovnávání paketu do předchozího řetězce nebo ukončit porovnávání paketu jeho zahozením či akceptováním. Každý řetězec pravidel má také nastavenou akci, která se s paketem provede v případě, že se nanalezne shoda ani s jedním z pravidel v řetězci.

3.2.1 NETFILTER

Netfilter při zpracování paketů využívá 3 hlavní kategorie [8]:

- INPUT
- FORWARD
- OUTPUT

Při příchodu paketu mohou nastat 2 situace. Buďto může být adresovaný do firewallu (INPUT) nebo ho firewall přepošle (FORWARD). Firewall se rozhoduje co udělat s paketem podle předem definovaných pravidel.

Pravidla mohou zahrnovat například:

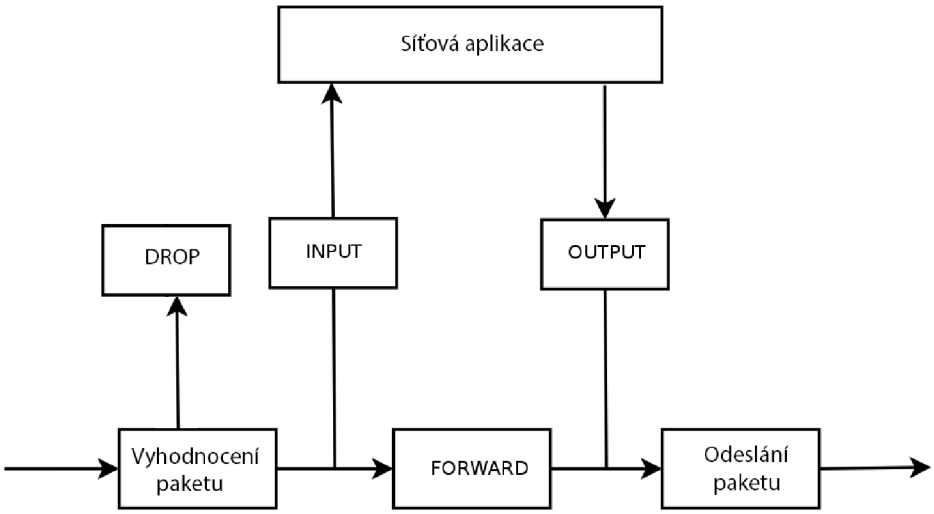
- rozhraní na který paket přišel
- rozhraní kterým paket odejde
- jaký využívá paket protokol na L4
- zdrojovou IP adresu nebo rozsah IP adres
- cílovou IP adresu nebo jejich rozsah

Pokud je paket adresován do zařízení a vyhovuje všem pravidlům, je předán dále systému ke zpracování (INPUT).

V případě že vyhovuje pravidlům pro přeposlání (FORWARD), potom ho firewall přepošle určeným portem dále.

V případě že paket nevyhovuje pravidlům, firewall ho nepustí dále. V takovém případě je několik možností co s paketem udělat. Je možné ho například zahodit a dále se jím nezabývat. Nebo můžeme odesílateli odpovědět že například do systému nemá přístup. Nebo při autorizacích se odešle příslušná odpověď odesílateli.

Vzhledem k tomu, že pravidla mohou být pro některé případy příliš jednoduchá, máme možnost je řetězit. Aby paket nebyl zahozen, potom musí splňovat všechna pravidla která jsou v řetězu.



Obrázek 3.1: Průchod paketu netfilterem.

3.3 Firewall ve FreeBSD

FreeBSD využívá pro kontrolu síťového provozu tři druhy firewallu (PF, IPFW a IPFILTER) a dva druhy straffic shaperu (ALTQ a DUMMYNET) pro kontrolu velikosti pásma [3]. Každý z těchto firewallů funguje odlišně a využívá jinou syntaxi pravidel.

3.3.1 PF

Packet Filter je systém pro filtrování TCP/IP provozu a překlad síťových adres. Je schopný normalizovat TCP/IP provoz, kontrolovat šířku pásma a priorizovat pakety.

3.3.2 IPFW

IPFW je stavový firewall, který podporuje IPv4 i IPv6. Paket je při příchodu porovnáván s jednotlivými pravidly a ve chvíli, kdy je nalezena shoda s pravidlem, je pro vedena příslušná akce. Znamená to že vždy první pravidlo vyhrává. V případě, že není nalezena shoda, IPFW v tichosti zahazuje paket.

3.3.3 IPFILTER

IPF funguje na rozdíl od IPFW tak, že vyhrává poslední pravidlo, které obsahuje shodu. Takže i v případě, že první pravidlo, u kterého se našla shoda povoluje paketu projít a dále se najde pravidlo, které ho zablokuje, bude paket zahozen. V případě IPFW by byl paket díky první shodě propuštěn.

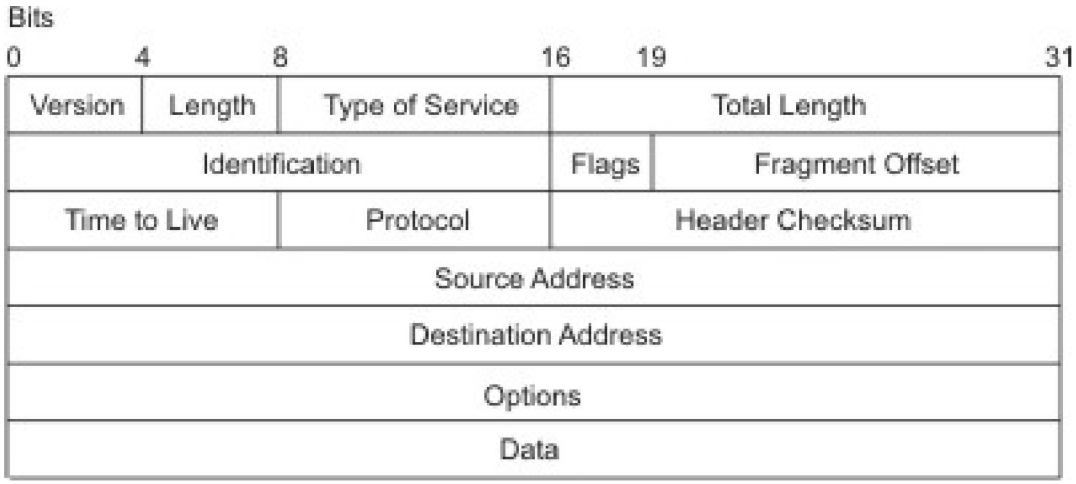
Kapitola 4

Zpracování jednotlivých protokolů

4.1 IP

IP protokol zajišťuje posílání dat mezi počítači na Internetu. Tento protokol funguje na L3 vrstvě OSI modelu. Každý počítač má v síti alespoň jednu unikátní IP adresu, podle které je možné ho identifikovat. Pakety, které jsou posílány po síti pak obsahují jak IP adresu přijemce, tak i IP odesílatele. Obrázek IP hlavičky paketu je na obrázku 4.1.

V dnešní době jsou nejvíce využívané IPv4 (32 bitů) adresy, ale kvůli jejich vyčerpání se začaly využívat také IPv6 adresy (128 bitů), které jsou delší a díky tomu poskytují více adresního prostoru [9]. Tyto adresy obsahují také masku sítě, která mumožňuje sdružovat je do podsítí, což usnadňuje směrování. IP adresy mohou být také statické nebo dynamické. Statická adresa se nemění a je v síti přidělena vždy stejnému zařízení. Dynamická adresa je dočasná adresa, která může být v síti přiřazena kterémukoliv zařízení připojenému do sítě.



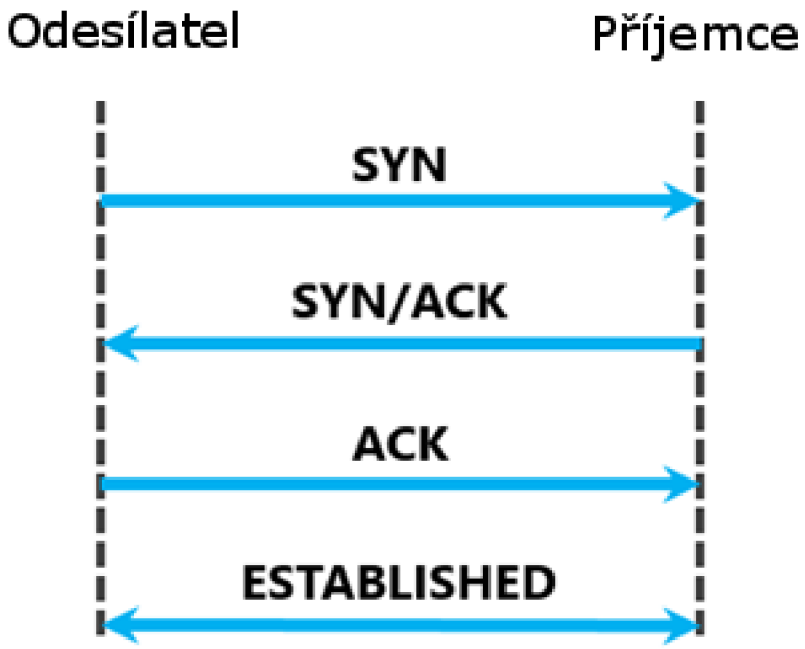
Obrázek 4.1: IP hlavička paketu

4.2 TCP

Jedna zpráva bývá většinou rozdělena na více paketů. Tyto pakety nemusí být vždy poslány k cíli stejnou cestou. Díky tomu nedorazí do cíle ve stejném pořadí, v jakém byly odeslány. Proto se využívá protokol TCP (Transmission Control Protocol), který má za

úkol zabezpečit spolehlivé doručení paketů ve správném pořadí. Tento protokol funguje na L4 vrstvě OSI modelu a umožňuje programům vytvořit a udržovat spojení, dokud nejsou zprávy odeslány. Při rozdělení zprávy na pakety jsou tyto pakety protokolem TCP očíslovány a následně odeslány IP protokolem. V případě, že strana příjemce neobdrží všechny pakety, odešle požadavek odesílateli o znovuposlání chybějících paketů.

Pro navázání spojení využívá protokol TCP tzv. handshake [15]. Jde o odeslání tří zpráv mezi odesílatelem a příjemcem (obr. 4.2). Odesílatel odešle synchronizační zprávu SYN. Příjemce obdrží zprávu SYN a následně odešle potvrzení o přijetí této zprávy SYN-ACK. Odesílatel po přijetí potvrzení synchronizace odešle potvrzení spojení ACK a naváže toto spojení.



Obrázek 4.2: Handshake u TCP protokolu

Protokol TCP má tedy za úkol poskládat pakety do správného pořadí a zajistit, aby byli doručeny všechny pakety právy. Toto chování nemusí být ovšem vždy žádoucí. Některé aplikace, například volání přes internet (VoIP), nevyžadují, aby byly doručeny všechny pakety, neboť by to zdržovalo jejich chod. V takových případech se využívá protokol UDP, který ztrátu paketů toleruje.

4.3 UDP

UDP (User Datagram Protocol) je alternativa k TCP protokolu, která ovšem toleruje ztrátu paketů a zaměřuje se především na co nejmenší latenci. Stejně jako TCP funguje na L4 vrstvě OSI modelu. UDP tedy při posílání paketů nestará o jejich číslování. Při ztrátě některých paketů se neodesílá požadavek o jejich opětovné odeslání. Díky tomu je přenos

paketů rychlejší než u TCP protokolu, díky čemuž se UDP využívá při videohovorech, volání přes Internet nebo při hraní her, kde nevadí, když jsou některé pakety ztraceny.

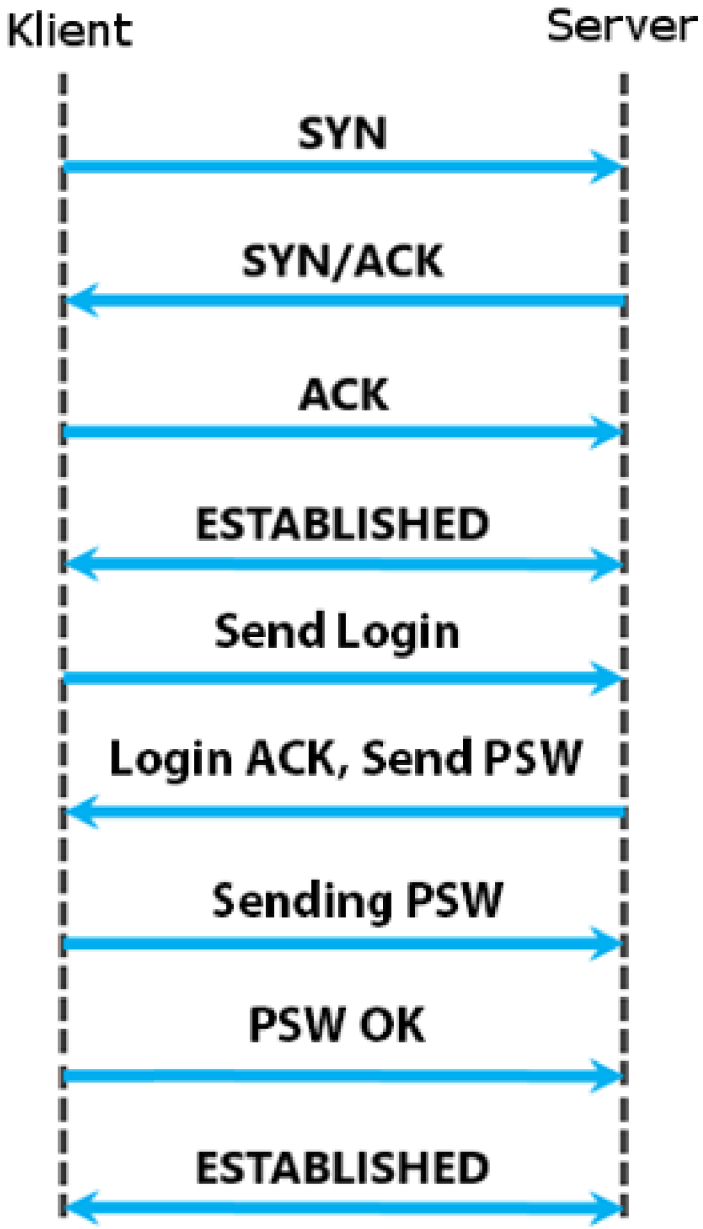
UDP mohou ale využívat i aplikace, které vyžadují doručení všech paketů. V takovém případě musí ale kontrolu doručení všech paketů a jejich poskládání do správného pořadí obstarat sama aplikace [16]. Toto je vhodné u aplikací, které přenáší velké soubory, neboť UDP je rychlejší než TCP.

4.4 FTP

FTP (File Transfer Protocol) slouží primárně k přenosu dat. FTP protokol využívá při přenosu dva kanály. Jeden kanál slouží pro komunikaci při přenosu dat a druhý pro přenos dat samotných. Díky tomuto protokolu může klient na server nahrávat soubory, stahovat z něj soubory, přejmenovávat je, kopírovat apod. FTP může fungovat ve dvou módech (aktivní a pasivní). V aktivním módu se po požadavku klienta na server otevře kanál pro přenos dat a začne se vysílat datový tok. V pasivním módu zase server využívá komunikační kanál pro odeslání dat klientovi potřebných pro otevření datového kanálu.

Uživatel může využívat FTP protokol buďto pomocí příkazové řádky, nobo pomocí programů určených právě pro tuto činnost. Z bezpečnostních důvodů byl FTP protokol mnohokrát rozšířen pro dosažení větší bezpečnosti (SFTP, FTPS) [2].

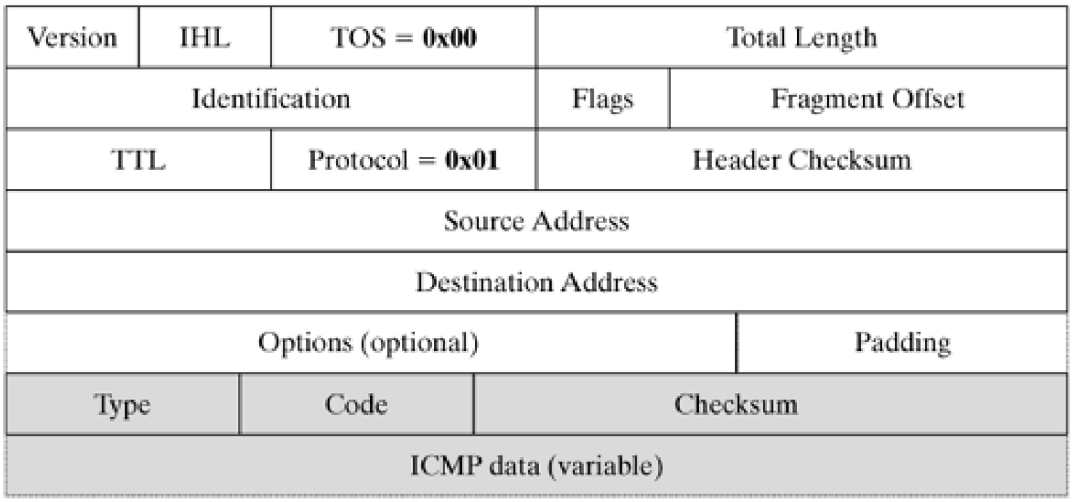
Navázání komunikace mezi klientem a serverem probíhá následovně. Klien odešle na server požadavek SYN. Server po přijetí požadavku SYN odešle odpověď SYN-ACK. Klient poté potvrdí přijetí potvrzení požadavku a odešle na server potvrzení ACK. Po navázání spojení odešle klient na server svůj login. Server po přijetí platného loginu odešle klientovi potvrzení loginu a požadavek na heslo. Klient poté odešle své heslo. V případě, že je heslo správné může začít přenos požadovaných souborů. Tento proces je také znázorněn na obrázku 4.3



Obrázek 4.3: Handshake u FTP protokolu

4.5 ICMP

Zprávy protokolu ICMP tvoří při filtrování paketů zvláštní skupinu, u které se pakety nedají pouze zahodit nebo přeposlat. V případě filtrování tímto způsobem by byla negativně ovlivněna komunikace. Naopak pouhé přeposílání ICMP zpráv bez jejich kontroly by znamenalo bezpečnostní riziko. Proto firewall musí obsahovat sadu speciálních pravidel pro tento protokol. Ukázka hlavičky zpráv ICMP je na obrázku 4.4.



Obrázek 4.4: Ukázka hlavičky ICMP.

4.5.1 Druhy ICMP zpráv

Protokol ICMP má velké množství zpráv, které se dají rozdělit do tří hlavních kategorií [7]. Tyto kategorie jsou popsány níže.

Chybové zprávy

Zprávy, které jsou poslány odesílateli v případě, že paket nemohl být doručen. Existují čtyři různé chybové správy, každá s různými podtypy.

Kontrola spojení

Monitorování spojení pomocí echo dotazů a odpovědí, které využívají příkazy ping a traceroute.

Objevovací funkce

- Vyhledávání sousedních zařízení a určování jejich IP adres a adres síťové vrstvy. Tyto zprávy také slouží ke kontrole duplicity adres. Zde jsou čtyři druhy zpráv:
  - Neighbor Solicitation(NS)
  - Neighbor Advertisement(NA)
  - Router Solicitation(NA)
  - Router Advertisement(NA)
- Kontrola, zda je sousední zařízení stále dosažitelné se stejnými adresami, s jakými bylo objeveno (Neighbor Unreachability Discovery - NUD) a informování sousedů o změnách adres linkové vrstvy.
- Vyhledávání routerů a zjišťování, jak je možné získat IP adresu pro připojení do podsítě.
- Zjištění prefixů a jiných nastavení (včetně doporučeného počtu skoků) od routeru.

- Při využívání SEcure Neighbor Discovery (SEND) pro autentizaci připojených routerů.

4.5.2 Filtrování ICMP paketů

ICMP pakety se při filtrování dělí na dvě základní skupiny:

- pakety, které mají přes firewall projít
- pakety, které jsou adresované na rozhraní firewallu

Dále si pakety rozdělíme do kategorií:

- Nesmí být zahozeny
- Neměly by být zahozeny
- Zprávy, které mohou být zahozeny firewallem, protože by je koncové zařízení stejně zahodilo.
- Zprávy, které nemají být zahozeny kvůli lokálním pravidlům

Následuje seznam ICMP zpráv, které by měly být zahozeny, neměly by být zahozeny nebo nesmí být zahozeny[14].

4.5.3 Pakety, které mají projít přes firewall

Pakety, které nesmí být zahozeny

Chybové zprávy potřebné k navázání a udržení spojení:

- Destination Unreachable (Typ 1) - všechny kódy
- Packet Too Big (Typ 2)

Zprávy pro kontrolu spojení:

- Echo Request (Typ 128)
- Echo Response (Typ 129)

Pakety, které by obvykle neměly být zahozeny

Chybové zprávy:

- Time Exceed (typ 3) - kód 1
- Parameter Problem (Typ 4) - kód 0

Mobilní IPv6 zprávy:

- Home Agent Address Discovery Request (Typ 144)
- Home Agent Address Discovery Reply (Typ 145)

Pakety, které by byly stejně zahozeny

Address Configuration a Router Selection zprávy (musí mít nastavený limit skoků na 255):

- Router Solicitation (Typ 133)
- Router Advertisement (Typ 134)

Link-local multicastové oznamovací zprávy (musí mít lokální zdrojovou adresu):

- Listener Query (Typ 130)
- Listener Report (Typ 131)

SEND Certificate Path oznamovací zprávy (musí mít nastavený limit skoků na 255):

- Certificate Path Solicitation (Typ 148)
- Certificate Path Advertisement (Typ 149)

Multicast Router Discovery zprávy (musí mít lokální zdrojovou adresu a limit skoků 1):

- Multicast Router Advertisement (Typ 151)
- Multicast Router Solicitation (Typ 152)

4.5.4 Pakety adresované na rozhraní firewallu

Pakety, které nesmí být zahozeny

Chybové zprávy potřebné k navázání a udržení spojení:

- Destination Unreachable (Typ 1) - všechny kódy
- Packet Too Big (Typ 2)

Zprávy pro kontrolu spojení:

- Echo Request (Typ 128)
- Echo Response (Typ 129)

Address Configuration a Router Selection zprávy:

- Router Solicitation (Typ 133)
- Router Advertisement (Typ 134)

Link-local multicastové oznamovací zprávy (musí mít lokální zdrojovou adresu):

- Listener Query (Typ 130)
- Listener Report (Typ 131)

SEND Certificate Path oznamovací zprávy:

- Certificate Path Solicitation (Typ 148)
- Certificate Path Advertisement (Typ 149)

Multicast Router Discovery zprávy:

- Multicast Router Advertisement (Typ 151)
- Multicast Router Solicitation (Typ 152)

Pakety, které by obvykle neměly být zahozeny

Chybové zprávy:

- Time Exceed (typ 3) - kód 1
- Parameter Problem (Typ 4) - kód 0

Pakety, které by byly stejně zahozeny

Router Renumbering zprávy:

- Router Renumbering (Typ 138)

Mobilní IPv6 zprávy:

- Home Agent Address Discovery Request (Typ 144)
- Home Agent Address Discovery Reply (Typ 145)

Zprávy experimentálního Seamoby protokolu:

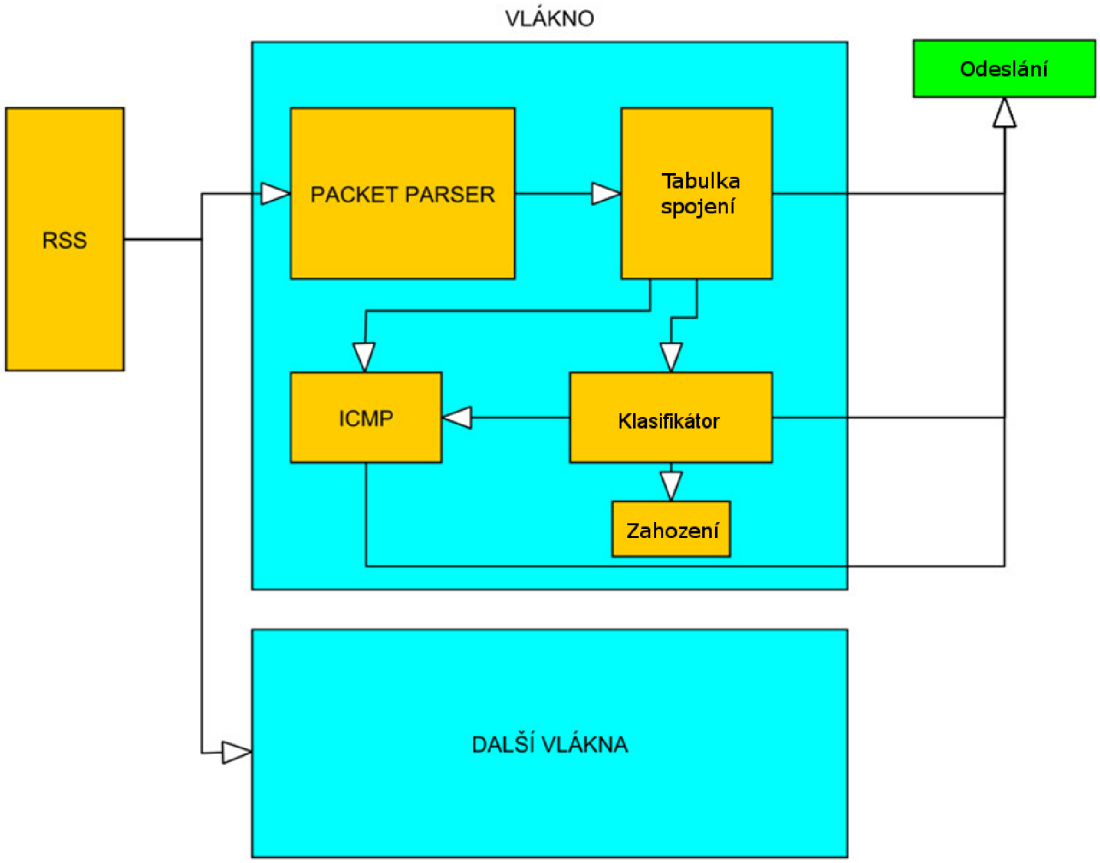
- Seamoby Experimental (Typ 150)

Kapitola 5

Návrh firewallu

Firewall se bude skládat ze tří hlavních částí (obr. 5.1): packet parseru, classificatoru a flow cache. Tyto tři části bude obsahovat každé samostatné vlákno. Rozdělování paketů do vláken bude zajišťovat jádro systému pomocí RSS, která podle IP adresy určí, které vlákno paket zpracuje. Poté, co bude paket přidělen určitému vláknu, firewall vyhodnotí, co se s ním stane. V případě, že paket bude splňovat podmínky pro průchod firewallem, bude přeposlán dále. Pokud podmínky splňovat nebude, bude zahozen. V případě zahození ICMP paketů bude odeslána ke zdroji zpráva o zahození paketu.

Firewall bude rozdělen do více vláken kvůli zvýšení rychlosti. Každé vlákno bude zpracovávat určitý druh paketů. O rozdělení těchto paketů se postará RSS. Všechna vlákna budou obsahovat stejné komponenty (Packet Parser, ICMP, Classificator, Flow Cache) a měly by obsahovat také stejná pravidla pro vyhodnocování paketů.



Obrázek 5.1: Návrh firewallu

5.1 PACKET PARSER

Tato část firewallu zjistí z hlavičky paketu potřebná data, jako jsou zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port a použitý protokol. Poté se tato data použijí k dalšímu zpracování.

5.2 FLOW CACHE

Flow cache je v podstatě tabulka záznamů o již zpracovaných spojeních. V případě zahození nebo projití paketu se o něm ve flow cache uchová záznam, aby další pakety náležící ke stejnému spojení mohly být přeposlány nebo zahozeny bez vyhodnocování. Flow cache bude tedy tabulka spojení, ve které se budou po určitou dobu uchovávat záznamy. Je potřeba jednou za určitou dobu flow cache projít a zkontrolovat. Pokud některým záznamům nevypršela platnost, je třeba vymazat tyto záznamy. Tabulku bude také potřeba projít při změně pravidel a v případě, že nějaký záznam nevyhovuje novému pravidlu, ho také smazat.

5.3 CLASSIFICATOR

Classificator má za úkol postarat se o pakety, které nemají záznam ve flow cache tabulce. podle předem definovaných pravidel vyhodnotí, co se má s příchozím paketem stát. V případě, že paket nebude vyhovovat některému z pravidel, bude zahozen. V případě, že bude vyhovovat pravidlům, bude firewallem přeposlán dále. U obou akcí se záznam o paketu zanese do flow cache, aby se stejné příchozí pakety nemusely znovu vyhodnocovat, protože proces vyhodnocení v klasifikátoru by měl být časově náročnější než proces vyhodnocení pomocí flow cache.

5.4 ICMP

Pakety protokolu ICMP tvoří zvláštní skupinu, díky čemuž se nedají jen tak zahodit. Většinou je potřeba i při zahození paketu odeslat jeho odesílateli zprávu o zahození. Díky tomu se k ICMP paketům musí přistupovat jinak než k ostatním paketům, které se prostě zahodí.

Kapitola 6

Popis DPDK

Pakety jsou na počítačích zpracovávány jádrem OS. Toto zpracování ovšem nemusí být vždy dostatečně rychlé. Proto pro dosažení vyšší rychlosti síťových aplikací, je vhodné zpracovávat pakety v uživatelském prostoru systému a ne pomocí jádra. K minimalizaci režie jádra se dá využít například knihovna DPDK (Data Plane Development Kit). Ta je tvořena z několika částí, jako je EAL, poll-mode ovladače a dalších knihoven, které jsou popsány dále v textu.

6.1 EAL

DPDK využívá EAL (The Enviroment Abstraction Layer) k zajištění přístupu k základním zdrojům (hardware, paměť) a komunikaci s operačním systémem. Dalšími funkcemi může být například rozpoznání hardwaru, správa procesů nebo zpracování přerušení [1].

6.1.1 Zpracování přerušení

EAL umožňuje zaregistrovat funkce pro zpracování přerušení, které mohou být vyvolány z vnitřních funkcí DPDK. Příkladem takovýchto přerušení mohou být například přerušení vyvolaná uživatelem, přerušení příchozího toku dat nebo odpojení kabelu.

6.1.2 Zjištění funkcí CPU

Je možné během běhu programu také zjistit, jaké funkce poskytuje CPU. K tomu se využívá funkce `rte_cpu_get_feature()` [1]. Díky tomuto nástroji je možné rozhodnout, zda bude aplikace vůbec spuštěna, protože různé aplikace mohou mít různé nároky na hardware.

6.1.3 Rezervace operační paměti

EAL také zajišťuje rezervaci paměti, neboť některé aplikace mohou mít vysoké paměťové nároky. K tomu se využívají takzvané hugepages. To jsou speciální stránky paměti, které se obvykle využívají u aplikací s velkou náročností na operační paměť. Obvykle mají velikost 2MB, ale mohou být i větší. Po rezervaci je tato paměť dostupná pro ostatní knihovny DPDK.

6.1.4 Práce s vlákny

Velkou výhodou EALu je možnost spuštění vláken na začátku programu, což nám velice zjednoduší práci s nimi. Nemusíme se totiž starat o jejich spouštění a ukončování. Při využití této funkcionality je spuštěné hlavní vlákno, které se nazývá master, ve kterém je inicializován program a struktury. Ostatní vlákna jsou dále označena jako slave.

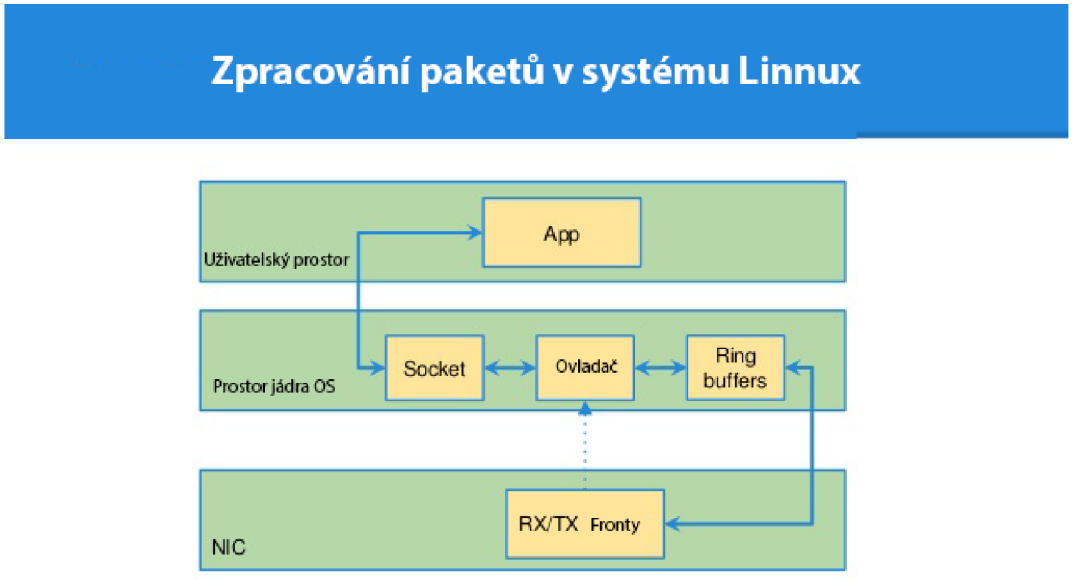
EAL rozpozná počet dostupných procesorů a jader, na jehož základě je odvozena hodnota *lcores* (logical cores), což je počet jader procesoru poskytnutý aplikaci, který se nastavuje pomocí argumentů programu. Ke každému jádru může být přiřazeno jedno vlákno.

6.2 Ovladače síťových karet

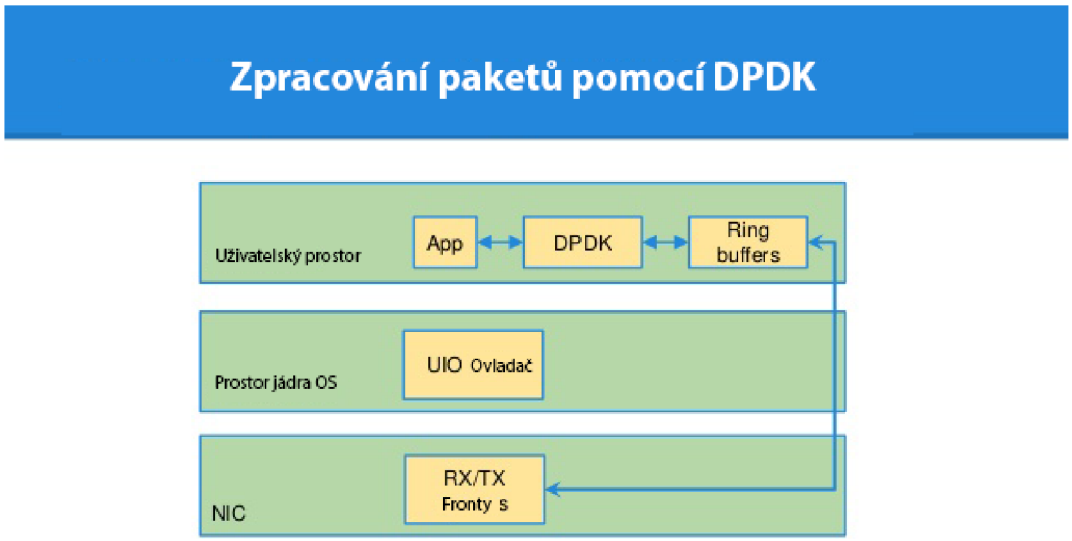
Jedním z účelů knihovny DPDK je zrychlit zpracování paketů oproti klasickému zpracování pomocí jádra OS. Aby toho mohlo být dosaženo, je třeba, aby příchozí pakety byly doručeny přímo DPDK aplikaci (obr. 6.2). Z toho důvodu je potřeba, aby DPDK přistupovalo přímo k síťovému adaptéru. K tomu slouží v DPDK API pro tvorbu ovladačů síťových adaptérů. V DPDK jsou taktéž již přibaleny ovladače k některým síťovým adaptérům [13].

Rozdíl přístupu těchto ovladačů oproti klasickému přístupu (obr. 6.1) OS je ten, že se při příchodu paketu nevyvolá přerušení a obslužní rutinu, ale na paket se čeká v aktivní smyčce. Tento přístup je efektivnější z pohledu latence.

Ke každému síťovému rozhraní, které je využívané DPDK aplikací, se přiřadí jeden z DPDK ovladačů. Toto ovšem znemožní využívání těchto adaptérů v rámci OS, ale jsou dostupné pouze DPDK aplikaci.



Obrázek 6.1: Zpracování paketů pomocí DPDK



Obrázek 6.2: Zpracování paketů v systému Linux

6.3 Mempool knihovna

EAL inicializuje paměť, která může být tvořena hugepages. Pro přistup k této paměti je třeba použít API, které je součástí knihovny `rte_memzone`. Tato knihovna umožňuje přistupovat právě k paměti, kterou na začátku spuštění aplikace alokoval EAL. I když lze použít funkce této knihovny přímo v aplikaci, není to vždy třeba, neboť knihovny, které pracují s alokací paměti, mají využití knihovny `rte_memzone` v sobě již implementované. Příkladem může být například knihovna Mbuf, která je popsána níže.

Mempool knihovna se využívá k alokaci datových struktur o předem známé velikosti a typu. Volné struktury jsou uchovány v paměti, dokud nejsou použity. Použité struktury je možné do mempoolu opět vrátit, aby mohly být znovu použity. Při inicializaci se nastavuje velikost a počet objektů, které je možné z mempoolu získat.

Je zde také možnost nastavit specifické zarovnání objektů v paměti, čímž lze docílit větší výkonosti. Cílem tohoto zarovnání je co nejefektivněji využít kanály DRAM [13].

6.4 Mbuf knihovna

Knihovna mbuf slouží k vytvoření datových struktur, které DPDK používá k uložení příchozích paketů. Tyto zásobníky jsou uloženy v mempool za pomocí Mempool knihovny. Paket je uchováván v celku jako struktura , ve které jsou uchovávána jak metadata, tak data paketu. Metadata obsahují informace jako typ zprávy, velikost, offset na začátek dat a ukazatel na další struktury mbufu, který umožňuje zřetězení. To se využívá u velkých paketů, u kterých je potřeba zřetězení struktur mbufu.

Knihovna mbuf nám umožňuje provádět s daty tyto operace [11]:

- Zjištění délky dat
- Získání ukazatele na začátek dat
- Přidání dat před stávající data
- Připojení dat za stávající data
- Odebrání dat ze začátku zásovníku
- Odebrání dat z konce zásobníku

6.5 Hash knihovna

DPDK umožňuje využití hash knihovny, která umožňuje pomocí klíče vyhledat záznamy vstupů v tabulce. Tato délka je nastavena při vytváření hash tabulky. Hash knihovna podporuje různé funkce, jako jsou přidání záznamu, odebrání záznamu a vyhledání záznamu [4].

6.5.1 Přidání záznamu

Při přidávání záznamu do tabulky, je potřeba zadat také klíč k záznamu. V případě, že je klíč přidán, nebo tabulka již obsahuje záznam s tímto klíčem, je vrácena pozice záznamu. V případě, že se záznam nepodařilo uložit, je vrácena záporná hodnota.

6.5.2 Odstranění záznamu

Záznam, který chceme odstranit, se vyhledá podle klíče. V případě, že je nalezen záznam se shodným klíčem, je tento záznam odstraněn a funkce vráti pozici, na které byl záznam nalezen. Pokud nebyl nalezen záznam se shodným klíčem, je vrácena záporná hodnota.

6.5.3 Vyhledání záznamu

K vyhledání záznamu slouží klíč. V případě shody je vrácena pozice nalezeného záznamu. Pokud nebyla nalezena shoda, je vrácena záporná hodnota.

Hash tabulka také umožňuje vyhledávat, přidávat nebo mazat záznamy s předem vypočítaným hashem. V takovém případě se poskytne právě tento hash a klíč k záznamu. Toto vylepšení slouží ke zvýšení rychlosti vyhledávání, protože se nemusí znovu počítat hash.

Dále je možné uložit do záznamu i vlastní data, která jsou vložena do tabulky spolu s klíčem. Tato data mohou mít ale maximální velikost 8 byte. V případě, že jsou data větší, musíme si uložit pouze ukazatel na ně.

6.6 Klasifikace paketů

DPDK obsahuje ACL knihovnu, která umožňuje klasifikaci paketů podle předem určených pravidel. Umožňuje clasifikaci podle pravidel v různých kategoriích a vyhledání nejlepší shody (pravidlo s nejvyšší prioritou) v každé kategorii.

Tato knihovna umožňuje [12]:

- Vytvoření nového pravidla
- Přidání pravidla k již používaným pravidlům
- Vytvoření dočasné struktury pro každé pravidlo, která je potřebná ke klasifikaci paketů
- Klasifikaci paketů podle pravidel
- Smazání pravidel, jejich struktur a uvolnění alokované paměti

6.6.1 Definice pravidel

ACL knihovna umožňuje uživateli vytvářet vlastní pravidla. Z důvodu zvýšení rychlosti vyhledávání mezi pravidly je první pole pravidla vždy dlouhé jeden byte. Ostatní pole jsou ve skupinách po čtyřech bytech.

Pro definici každého pole pravidla se využívá struktura obsahující:

- *type* - může obsahovat hodnoty [12]:
  - _MASK - pro pole jako jsou IP adresy, které mají hodnotu a masku definující počet důležitých bitů
  - _RANGE - pro pole jako porty, které mají nižší a vyšší hodnotu
  - _BITMASK - pro pole identifikující protokool, které mají hodnotu a bitovou masku
- *size* - definuje délku pole v bajtech.
- *field_index* - reprezentuje pozici pole v rámci pravidla. Nabývá hodnoty 0 až N-1 pro N polí.
- *input_index* - specifikuje, ke které skupině polí patří. Důvodem je seskupování polí po čtyřech bajtech.
- *offset* - ukazuje na začátek pole.

Při vytváření pravidel musíme také zadat prioritu pravidla, masku kategorie a data, která mají být vrácena v případě nalezení shody s pravidlem. Pokud není nalezena shoda, je vrácena nula.

Kapitola 7

Zpracování firewallu pomocí DPDK

V rámci vypracování této diplomové práce byla vypracována i implementace firewallu s pomocí DPDK knihovny. Tato aplikace má za úkol pomocí předem daných směrovacích pravidel rozhodovat o předávání nebo zahazování příchozích paketů.

7.0.1 Základmí architektura

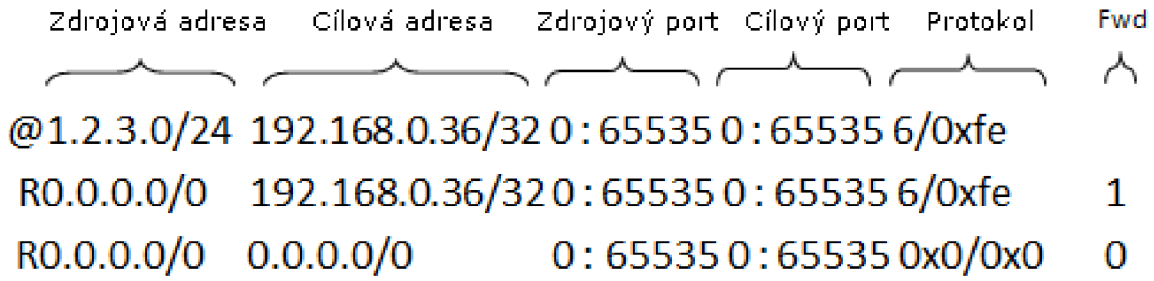
Principem fungování aplikace je rozdělení zpracování paketů mezi více vláken. Pakety přicházející na rozhraní jsou rozdělovány mezi jednotlivá vlákna a dále vyhodnocovány. V případě, že vyhovují pravidlům pro přeposlání, jsou přesměrovány na příslušný port. V případě, že nevyhovují pravidlům pro přeposlání, jsou zahozeny.

Program se skládá ze dvou hlavních částí. Společné části, kde se inicializuje EAL a všechny potřebné struktury a z části pro samostatná vlákna.

7.0.2 Společná část

Nejprve se inicializuje EAL za pomocí příkazů, které se předávají pomocí argumentů při spuštění programu.

Následně se inicializují pravidla pro klasifikaci. Tato pravidla jsou uložena v souborech *ipv4rules.db* pro IPv4 a *ipv6rules.db* pro IPv6. Pravidlo, které začíná znakem @ označují pravidla pro zahození paketu. Pokud paket odpovídá tomuto pravidlu, bude zahozen. Pravidla začínající znakem *R* jsou směrovací pravidla. U těchto pravidel je na konci uveden port, na který má být paket přesměrován. Ukázka záposu pravidel je na obrázku 7.1.



Obrázek 7.1: Příklad pravidel pro firewall

Poté je třeba ke každému portu přiřadit vstupní a výstupní fronty. Každé vlákno programu musí mít na každém portu svoji vstupní a výstupní frontu. Jako identifikátor těchto front je použito číslo vlákna, které nabývá hodnot 0 - N-1, kde N je počet vláken. Celkový počet vláken zjistíme pomocí hodnity *lcores* kterou zjistíme pomocí EALu. Každé vláknjo poté bude přijímat a odesílat pakety pouze pocí své vstupní a výstupní fronty.

Po přiřazení front se pomocí funkce `rte_eal_mp_remote_launch` spustí pro každé vlákno hlavní smyčka programu, která zpracovává pakety.

7.0.3 Zpracování paketů

Každé vlákno získává pakety z fronty rozhraní pomocí funkce `rte_eth_rx_burst`. Tato funkce čte pakety po dávkách, přičemž maximální velikost dávky je defaultně 32 paketů. Pakety se uloží do struktury *mbuf*.

Pro implementaci flow cache byla zvolena hash tabulka. Kvůli možnosti využití stejné struktury pro IPv4 i IPv6 protokoly je délka všech klíčů stejná. U každého pakeru je díky nastavení RSS předem vygenerovaný hash. Tento hash se využívá v hash tabulce jako klíč k záznamům jednotlivých toků. U každého paketu se prověří, zda nemá již záznam v hash tabulce. V případě, že má, provede se akce uložená v tabulce (zahození nebo přeposlání). Pokud paket v tabulce nemá záznam, je zpracován dále pomocí ACL klasifikace.

Následně se provede extrakce a zpracování dat hlaviček paketů, které neměly záznam v hash tabulce. Pro účely ACL potřebujeme zjistit především zdrojovou IP adresu, cílovou IP adresu, zdrojový port, cílový port a protokol. Tyto informace se spolu s ukazatelem na paket ve struktuře *mbuf* uloží do struktury `acl_search_t`.

Následně jsou pakety vyhodnocovány pomocí funkce rte_acl_classify. Tato funkce zajišťuje vyhledání shodného ACL pravidla pro pakety předané této funkci. U každého paketu najde první shodné pravidlo, podle kterého nastaví, zda má být paket přeposlán nebo zahozen. Pokud má být paket přeposlán, nastaví mu také port, ze kterého má být odeslán podle nalezeného pravidla.

Všechny vyhodnocené pakety se poté připraví k odeslání. Pokud paket nevyhovoval pravidlům, je zahozen a paměť, kterou zabíral je uvolněna. V případě, že paket vyhovoval pravidlům, je pomocí funkce `rte_eth_tx_buffer` vložen do výstupního zásobníku pro odpovídající port.

Každé vlákno jednou za dobu nastavenou pomocí proměnné *drain_tsc* vyprázdní celý výstupní zásobník pomocí funkce `rte_eth_tx_buffer_flush`.

U tohoto vícevláknového zpracování se ovšem vyskytli problémy, které se bohužel nepodařilo opravit. Při odesílání paketů se některé ztrácely, tudíž proggram odesílal méně paketů než přijímal. Vzhledem k nefunkčnosti tohoto řešení byla odstraněna i implementace hash tabulky, neboť nemohla být řádně otestována ani základní funkcionalita. Výše popsané řešení je tedy spíše teoretické a z větší části implementované, ale bohužel nefunkční. Z důvodu nefunkčnosti tohoto řešení je vypracována ještě druhá verze programu, která je popsána níže.

7.0.4 Verze pro jedno vlákno na port

Vzhledem k tomu, že se nepodařilo plně zprovoznit verzi aplikace, kde více vláken odebírá z jednoho portu pakety, byla vypracována verze, kde se o zpracování paketů stará pouze jedno vlákno.Tudíž za předpokladu, že aplikace bude obsluhovat dva porty, budou třeba pro její běh pouze jedno vlákno.

Běh aplikace je poté velice podobný jako ve verzi, kde více vláken obsluhuje jeden port. Pouze při inicializaci vstupních a výstupních front je přiřazena ke každému portu přiřazena pouze jedna vstupní a jedna výstupní fronta.

Po přiřazení front se spustí hlavní smyčka programu pouze pro jedno vlákno. Zde je téměř stejná funkcionalita jako ve verzi pro více vláken. Pakety se pomocí funkce `rte_eth_rx_burst` po dávkách čtou ze vstupní fronty. Po přečtení jsou pakety vyhodnoceny stejně jako u vícevláknové verze. Jen v této verzi není implementovaná hash tabulka toků.

Kapitola 8

Testování

Z důvodu nefunkčního vícevláknového řešení programu, nebylo možné otestovat jeho propustnost. Jak již bylo zmíněno, toto řešení odesílalo méně paketů, než přijímalo a to znemožnilo jakékoliv měření. Oproti tomu jednovláknová verze měla propustnost dostatečnou, a za předpokladu, že by se při použití více vláken škálovala výkonnost lineárně, by bylo možné dosáhnout pořadované rychlosti. Toto ovšem bohužel nebylo možné experimnetálně ověřit.

Problém při odesílání paketů byl nejspíše při jejich vkládání do zásobníku, který byl poté vyprázdněn pomocí funkce `rte_eth_tx_buffer_flush`. Při tomto vkládání je třeba zadat do které výstupní fronty má být paket vložen. Výstupní fronty jsou označené pomocí čísla vlákna, pro které jsou určené. Toto vkládání do výstupních front ale nejspíše nebylo provedeno správně, díky čemuž se pakety neuložily do správné fronty a poté nemohly být všechny odeslány.

Tento problém se bohužel nepodařilo vyřešit ani využitím funkce `rte_eth_tx_burst` namísto funkce `rte_eth_tx_buffer_flush`. V rámci tohoto řešení se pakety neukládají po jednom do výstupní fronty, ale ukládají se so struktury mbufu, která je poté odeslána v jedné dávce pomocí funkce `rte_eth_tx_burst`. Zde je třeba hlídat, kolik paketů bude v této dávce odesláno a tento počet předat funkci pomocí parametru. Hlídat počet paketů musíme z důvodu ACL. Ne všechny pakety, které přijdou budou totiž odeslány. Proto nemůžeme využít počet přijatých paketů, který nám vrátí funkce `rte_eth_rx_burst`.

8.1 Návrh na rozšíření

V rámci rozšíření programu by bylo možné, přidat funkci přidáváni a odebírání klasifikačních pravidel za běhu programu. K tomu by se dala využít ACL knihovna zabudovaná v DPDK. Tato knihovna totiž umožňuje přidání i odebrání pravidel. Za předpokladu implementace tohoto rozšíření by ovšem bylo třeba po každé změně pravidel projít záznamy o spojeních uložené v hash tabulce každého vlákna a upravit nebo vymazat záznamy, které by byly ovlivněny změněnými pravidly. Například při odstranění pravidla, které by zahazovalo pakety s určitým zdrojovým portem, musely být z hash tabulky odstraněny záznamy, které ovlivnilo toto pravidlo. Kdyby se tak nestalo, byly by i po odstranění pravidla zahazovány pakety, které pravidlo ovlivňovalo. Takto by se dělo až do vypršení platnosti záznamu v hash tabulce.

Další možností řešení problému se záznamy v tabulce spojení by bylo po přidání pravidla ji celou vyčistit. T oby ovšem na nějakou dobu po přidání pravidla zpomalilo provoz, neboť

by musely všechny pakety projít klasifikací. V případě tohoto řešení by ale byla jistota, že po přidání pravidla se projeví na celém provozu.

Pokud by testování ukázalo, že implementace hash tabulky není nutná, neboť klasifikace pomocí DPDK má dostatečnou rychlost, nebylo by nutné po změně pravidel už nic ošetřovat. Příchozí pakety by byly klasifikovány již podle nového pravidla. U nově přidaných pravidel by se musela nastavovat také priorita přidaného pravidla, neboť by mhla být v rozporu s již přidanými pravidly a tudíž by mohla negativné ovlivnit fungování firewallu nebo by se naopak nemusela vůbec projevit.

Kapitola 9

Závěr

Implementaci aplikace firewallu s rozdělením provozu do více vláken se bohužel nepodařilo plně zprovoznit, kvůli problémům se zahazováním paketů. Druhá verze programu, která ke zpracování paketů využívá pouze jedno vlákno je funkční. Její rychlost ale bohužel nedosahuje požadovaných rychlostí. Za předpokladu, že by se rychlost zpracování dat zvyšovala lineárně spolu s počtem jader, mělo by být možné dosáhnout požadované rychlosti.

Literatura

[1] Environment Abstraction Layer. DPDK [online]. [cit. 05.05.2017].
URL http://dpdk.org/doc/guides/prog_guide/env_abstraction_layer.html

[2] File Transfer Protocol (FTP) [online]. [cit. 18.01.2017].
URL http:
//searchenterprisewan.techtarget.com/definition/File-Transfer-Protocol

[3] Firewall in FreeBSD [online]. [cit. 10.01.2017].
URL http://www.freebsd.cz/doc/handbook/firewalls.html

[4] Hash Library. DPDK [online]. [cit. 06.05.2017].
URL http://dpdk.org/doc/guides/prog_guide/hash_lib.html

[5] How deep packet inspection works [online]. [cit. 25.11.2016].
URL http://www.wired.co.uk/article/how-deep-packet-inspection-works

[6] How Proxy Firewalls Work [online]. [cit. 25.11.2016].
URL http://www.bullguard.com/bullguard-security-center/pc-security/
computer-security-resources/how-proxy-firewalls-work.aspx

[7] ICMP Message Types [online]. [cit. 21.01.2017].
URL http://www.informit.com/articles/article.aspx?p=26557&seqNum=5

[8] Introduction to Netfilter [online]. [cit. 12.12.2016].
URL https://home.regit.org/netfilter-en/netfilter/

[9] IP address - Internet Protocol (IP) address [online]. [cit. 17.01.2017].
URL http://www.webopedia.com/TERM/I/IP_address.html

[10] Linux Firewall Tutorial: IPTables Tables, Chains, Rules Fundamentals [online]. [cit. 12.12.2016].
URL http://www.thegeekstuff.com/2011/01/iptables-fundamentals

[11] Mbuf Library. DPDK [online]. [cit. 06.05.2017].
URL http://dpdk.org/doc/guides/prog_guide/mbuf_lib.html

[12] Packet Classification and Access Control. DPDK [online]. [cit. 06.05.2017].
URL
http://dpdk.org/doc/guides/prog_guide/packet_classif_access_ctrl.html

[13] Poll Mode Driver. DPDK [online]. [cit. 06.05.2017].
URL http://dpdk.org/doc/guides/prog_guide/poll_mode_drv.html

[14] RFC4890 [online]. [cit. 22.01.2017].
URL https://www.ietf.org/rfc/rfc4890.txt

[15] TCP 3-Way Handshake (SYN,SYN-ACK,ACK) [online]. [cit. 17.01.2017].
URL
http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.shtml

[16] UDP (User Datagram Protocol) [online]. [cit. 18.01.2017].
URL http://searchmicroservices.techtarget.com/definition/UDP-User-Datagram-Protocol

[17] What are Windows Firewall and Windows Defender? [online]. [cit. 12.12.2016].
URL http://www.dummies.com/computers/computer-networking/network-security/what-are-windows-firewall-and-windows-defender/

[18] WHAT IS A FIREWALL? [online]. [cit. 25.11.2016].
URL https://www.paloaltonetworks.com/cyberpedia/what-is-a-firewall