VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

ANALÝZA PARAMETRŮ PRAVIDLOVÝCH SAD PRO KLASIFIKACI SÍŤOVÉHO PROVOZU

ANALYSIS OF PARAMETERS OF PACKET CLASSIFICATION RULE SETS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOZEF SABO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JIŘÍ MATOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Sabo Jozef**
Program: Informační technologie
Název: **Analýza parametrů pravidlových sad pro klasifikaci síťového provozu**
Analysis of Parameters of Packet Classification Rule Sets
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s architekturou TCP/IP, se síťovými protokoly IPv4 a IPv6 a s protokolem OpenFlow.
2. Nastudujte formát souborů parametrů používaných nástroji ClassBench a ClassBench-ng jako vstup pro generování umělých sad pravidel pro klasifikaci síťového provozu.
3. Navrhněte nástroj pro generování souborů parametrů ze sad pravidel pro klasifikaci síťového provozu popsaných v běžně používaných formátech.
4. Implementujte navržený nástroj.
5. Analyzujte dostupné sady pravidel pro klasifikaci síťového provozu a vygenerujte pro ně soubory parametrů ve formátu používaném nástroji ClassBench a ClassBench-ng.
6. Srovnejte výsledky analýzy s obdobnými analýzami popsanými v literatuře a diskutujte případné odlišnosti.
7. V závěru se zamyslete nad možnostmi dalšího rozšíření implementovaného nástroje.

Literatura:

- D. E. Taylor and J. S. Turner, "ClassBench: A Packet Classification Benchmark," in IEEE/ACM Transactions on Networking, vol. 15, no. 3, pp. 499-511, June 2007.
- J. Matoušek, G. Antichi, A. Lučanský, A. W. Moore, J. Kořenek, "ClassBench-ng: Recasting ClassBench After a Decade of Network Evolution," in 2017 ACM/IEEE Symposium on Architectures for Networking and Communications, IEEE CS, 2017, pp. 204-216.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

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

Vedoucí práce: **Matoušek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

Témou bakalarskej práce je analýza pravidiel používaných pre klasifikáciu paketov v počítačových sieťach. Ako klasifikácia paketov funguje a akú úlohu pri klasifikácií majú klasifikačné pravidlá je popísané v teoretickej časti práce. Formáty klasifikačných pravidiel používaných v reálnych nástrojoch v praxi sú v tejto časti opísané taktiež. Na základe týchto znalostí bol navrhnutý a implementovaný nástroj, ktorý umožňuje analyzovať položky klasifikačných pravidiel štandardnej IP pätice zo sád pravidiel s ľubovoľnými formátmi. Výstupom implementovaného nástroja je parametrový súbor, ktorého rôzne štatistiky a rozdelenia pravdepodobností skúmaných položiek pravidiel popisujú kompozíciu analyzovanej sady pravidiel. Tento parametrový súbor je možné použiť pre generovanie umelých sád pravidiel v nástrojoch ClassBench a ClassBench-ng. V záverečnej časti práce sú skúmané parametrové súbory vzniknuté implementovaným nástrojom z dostupných reálnych sád pravidiel.

Abstract

A theme of bachelor's thesis is an analysis of rules used for packet classification in computer networks. A theoretical part of the thesis introduces packet classification and describes the role of classification rules. This part also presents the format of classification rules utilized in real tools. Based on these information, a tool able to analyze IP 5-tuple classification rules in any format was designed and implemented. Output of the implemented tool is a parameter file containing different statistics and probability distributions of examined rule sets. This parameter file can be used for generating synthetic rule sets using ClassBench and ClassBench-ng tools. The final part of the thesis examines parameter files created by the implemented tool from available real rule sets.

Kľúčové slová

klasifikácia paketov, analýza klasifikačných pravidiel, ClassBench, ClassBench-ng

Keywords

packet classification, analysis of classification rules, ClassBench, ClassBench-ng

Citácia

SABO, Jozef. *Analýza parametrů pravidlových sad pro klasifikaci síťového provozu.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Matoušek, Ph.D.

Analýza parametrů pravidlových sad pro klasifikaci síťového provozu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jiřího Matouška, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jozef Sabo
15. mája 2019

Poďakovanie

Rád by som poďakoval Ing. Jiřímu Matouškovii, Ph.D., za jeho odborné vedenie, cenné rady pri tvorbe tejto práce a poskytnutie užitočných skriptov pre tvorbu grafov.

Obsah

Kapitola 1

Úvod

Klasifikácia paketov je dôležitá činnosť vykonávaná pri každej komunikácií na TCP/IP sieti. Jej úlohou je, čo najrýchlejšie roztriediť príchodzie pakety na rozhraní zariadenia s cieľom rozhodnúť o ich osude. Na zariadeniach, nazývaných smerovače, klasifikácia určuje, do ktorej siete alebo jej časti budú pakety ďalej smerované. Na iných zariadeniach, nazývaných firewally, je za účelom zvýšenia bezpečnosti na sieti klasifikácia použitá pre filtrovanie paketov prechádzajúcich cez ich rozhrania. V spomínaných aj vo všetkých ostatných zariadeniach vykonávajúcich klasifikáciu paketov má túto úlohu na starosti zariadením implementovaný klasifikačný algoritmus. Algoritmy rozdeľujú pakety z prichádzajúceho rozhrania do skupín s podobnými vlastnosťami podľa klasifikačných pravidiel uložených v databáze zariadenia. Rýchlosť fungovania algoritmu úzko závisí na množstve a zložitosti pravidiel. Keďže v súčasnej dobe sa počet a komplexnosť pravidiel v databázach zariadení stále zvyšuje, sú preto klasifikačné algoritmy aktuálnou témou rôznych výzkumov.

Užitočnými nástrojami pre testovanie klasifikačných algoritmov sú nástroje ClassBench [17] a ClassBench-ng [9]. Nástroje generujú umelú sadu filtrovacích pravidiel, ktorými potom testovaný algoritmus musí prechádzať, aby pre paket, prichádzajúci na rozhranie, našiel odpovedajúce pravidlo. Sada je generovaná na základe parametrového súboru, ktorý sa skladá zo štatistík a rozdelení pravdepodobností položiek pravidiel reálnej sady. Tieto štatistiky a rozdelenia pravdepodobností pravidiel zabezpečujú, že vygenerovaná sada sa bude podobať reálnej sade, ale nebude jej kópiou. Dôvodom, prečo sa pre testovanie klasifikačných algoritmov nepoužívajú reálne sady pravidiel, ale umelé vygenerované sady pravidiel je nedostupnosť reálnych sád. Organizácie nemôžu zverejniť klasifikačné (filtrovacie) pravidlá používané v ich firewalloch, pretože by to ohrozilo bezpečnosť na ich sieti.

ClassBench-ng podporuje tvorbu parametrových súborov, ale iba pre určitý formát reálnej sady pravidiel. ClassBench dokonca takúto podporu konvertovania reálnej sady do parametrového súboru v publikovanej verzii nemá vôbec. Témou tejto bakalárskej práce je navrhnúť a implementovať nástroj, ktorý dokáže vytvoriť parametrový súbor z akéhokoľvek formátu reálnej sady, ktorej pravidlá sú zložené z položiek štandardnej IP pätice. Kapitola 2 poskytuje teoretický úvod do sveta TCP/IP sietí. Klasifikácia paketov a generovanie umelých sád klasifikačných pravidiel pomocou nástrojov ClassBench a ClassBench-ng sú rozoberané v kapitole 3. Návrh nástroja pre konvertovanie reálnej sady ľubovoľného formátu do parametrového súboru opisuje kapitola 4. O implementácií navrhnutého nástroja je popísané v kapitole 5. V záverečnej kapitole 6 sú skúmané parametrové súbory vypočítané pomocou implementovaného nástroja z dostupných reálnych sád pravidiel.

Kapitola 2

Architektúra sietí typu TPC/IP

TCP/IP (Transmission Control Protocol and Internet Protocol) je označenie pre najpoužívanejšiu rodinu protokolov pre sprostredkovanie komunikácie medzi zariadeniami na rôznych typoch sietí. Pod pojmom protokol rozumieme dojednané pravidlá, ktorými sa riadia komunikujúce zariadenia. Funkcionalita TCP/IP je rozdelená do štyroch vrstiev. Na každej vrstve sa nachádza množina operujúcich protokolov. Výčet najdôležitejších protokolov na jednotlivých vrstvách je zobrazený v tabuľke 6.2. Komunikácia medzi dvoma koncovými uzlami (počítačmi) na sieti prebieha prostredníctvom protokolov v každej z vrstiev. V prechodových uzloch (smerovačoch) je implementovaná iba sieťová vrstva a vrstva sieťového rozhrania.

TCP/IP vrstva	TCP/IP protokoly
Aplikačná vrstva	HTTP, SMTP, DNS, DHCP
Transportná vrstva	TCP, UDP
Sieťová vrstva	IPv4, IPv6
Vrstva sieťového rozhrania	Ethernet

Tabuľka 2.1: Vrstvy TCP/IP s výčtom ich najdôležitejších protokolov

Najznámejším predstaviteľom TCP/IP typu siete je, celosvetový systém vzájomne prepojených počítačových sietí, Internet [21]. V súčasnosti je takáto architektúra skoro výhradne využívaná aj v sfére privátnych sietí. Dôvodom dominancie využívania sietí typu TCP/IP je ich flexibilita, robustnosť a nezávislosť na operačnom systéme [15].

2.1 Vrstva sieťového rozhrania

Najnižšia vrstva, vrstva sieťového rozhrania, nazývaná tiež ako linková vrstva, definuje ako sú dáta fyzicky vysielané a prijaté medzi susednými uzlami na sieti. Bloky dát, nazývané rámce, sú na linkovej vrstve pri prenose smerované na základe MAC adries uvedených v hlavičkách rámcov. MAC adresa je 48-bitový jednoznačný identifikátor sieťového rozhrania uvedený vo forme 12-miestneho hexadecimálneho čísla, napríklad 12:34:56:78:9A:BC [23]. Prvých 24 bitov adresy identifikuje výrobcu, na sieti komunikujúceho, zariadenia. Z uvedeného príkladu ide o čísla 12:34:56, ktoré identifikujú výrobcu zariadenia. Druhé 24 bity zápisu MAC adresy sú jedinečné pre daného výrobcu a jednoznačne identifikujú zariadenie

na sieti. Z uvedeného príkladu ide o čísla 78:9A:BC, ktoré identifikujú zariadenie. Linková vrstva zabezpečuje konvertovanie rámcov do formátov, ktoré sú prenášateľné na konkrétne použitom fyzickom médiu. Napríklad pri komunikácií po káblovanom médiu sú rámce konvertované do elektrických signálov [8]. Z toho vyplýva, že použitý protkol na vrstve závisí na konkrétne použitej prenosovej technológii.

2.2 Sieťová vrstva

Úlohou sieťovej vrstvy je vytvorenie logického spojenia medzi ľubovoľnými dvoma uzlami na sieti [16]. Najdôležitejším protokolom vrstvy je protokol IP (Internet Protocol), ktorého hlavnou úlohou je takéto spojenia vytvárať. Protokol prenáša sieťou dáta vo forme paketov opatrených IP hlavičkou, obsahujúcou IP adresu príjemcu a iné smerovacie informácie potrebné k doručeniu paketu. IP adresa slúži k jednoznačnej identifikácií uzlov na sieti. IP protokol sa nestará o spoľahlivosť, zahadzuje poškodené pakety. Ide mu hlavne, o čo najrýchlejší prenos dát. Pre zabezpečenie spoľahlivého prenosu sú k dispozícií protokoly vyšších vrstiev.

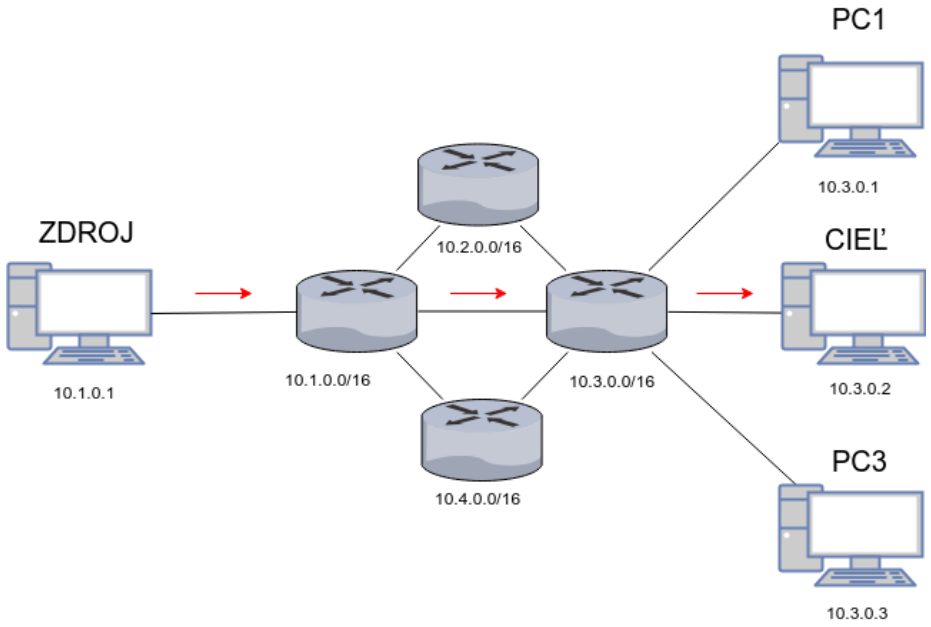
Proces zaslania paketu ľubovoľnému uzlu vyzerá na funkčnej sieti nasledovne:

1. Rozhodnutie o najvhodnejšom smere pre odoslanie paketu na základe IP adresy cieľa uvedenej v hlavičke paketu. Tomuto kroku sa hovorí smerovanie.
2. Po nájdení smeru sú pakety zabalené do rámcov a prostredníctvom linkovej vrstvy poslané najvhodnejšiemu susednému uzlu. Susedný uzol je identifikovaný na základe MAC adresy.
3. Vybraný susedný uzol si prijatý rámec rozbalí.
   (a) Ak je paket určený pre neho, predá ho vyšším vrstvám. Paket našiel svojeho príjemcu, proces zaslania paketu skončil úspechom.
   (b) Ak paket nie je určený pre neho, celý proces sa od bodu 1 pre uzol znova opakuje. Na uvedenom obrázku 2.1 sa celý proces opakuje až 3-krát pokiaľ paket nedorazí do cieľa. Spôsob prenosu IP protokolu je preto označovaný ako nespojovaný.

Protokol IP má dve verzie IPv4 a IPv6. Prvou používanou verziou bola verzia IPv4, na ktorej sa časom ukázalo, že množstvo adries 2^{32}, ktoré poskytuje, nie je dostačujúce, a tak vznikla verzia IPv6 s enormným zvýšením množstva adries na 2^{128}. Novšia verzia okrem spomínaného navýšenia rozsahu adries prišla aj s ďalšími výhodami ako: bezstavová automatická konfigurácia rozhraní, jednoduchší formát hlavičiek paketov a viac efektívne smerovanie [11]. Aj napriek týmto výhodám sa ešte stále častejšie pre adresovanie privátnych sietí používa staršia verzia IPv4, pretože poskytuje pre správu sietí menej náročný ľahko zapamätateľný formát adries. Verziami používané formáty IP adries a hlavičky paketov sú opísané v nasledujúcich sekciách.

2.2.1 Protokol IPv4

Verzia IPv4 pracuje s 32-bitovými adresami, ktoré sú zložené z 4 decimálnych čísiel s možnými hodnotami 0 až 255 oddelených bodkou, napríklad 10.0.0.1/24 [20]. Adresa sa skladá z adresy siete a adresy rozhrania. Adresa siete jednoznačne identifikuje sieť (informácia potrebná pre smerovanie paketov). Adresa rozhrania zas jednoznačne identifikuje uzol na danej



Obr. 2.1: Obrázok zobrazujúci prenos paketu zo zdrojového zariadenia s IP adresou 10.10.0.1 do cieľového zariadenia s IP adresou 10.3.0.2

sieti. Číslo za lomkou sa nazýva dĺžka prefixu. Dĺžka prefixu IPv4 adresy môže nadobúdať hodnoty z rozsahu 0 až 32. Hodnota dĺžky prefixu udáva počet bitov adresy patriacich adrese siete. Zvyšné bity adresy, na pozícii od hodnoty dĺžky prefixu až po 32 určujú adresu rozhrania. Takýto systém adresovania je známy pod skratkou CIDR (Classless Inter-Domain Routing). Alternatívnym spôsobom adresovania bolo, v minulosti používané, triedne adresovanie IP adries. IP adresy boli pri tomto spôsobe adresovania rozdelené do 5 tried (triedy A–E) a každá trieda mala pevne určenú dĺžku prefixu.

Hlavička IPv4 paketu je zobrazená na obrázku 2.2. Čísla v obrázku nad hlavičkou označujú veľkosti položiek hlavičky v bitoch. Jednotlivé položky hlavičky (okrem adries) sú stručne opísané v nasledujúcom zozname:

- **Verzia** značí verziu hlavičky paketu (IPv4 alebo IPv6).
- **IHL** obsahuje veľkosť hlavičky paketu. IPv6 hlavička nemá túto položku, pretože jej veľkosť je pevná (40 bajtov).
- **Typ služby** je položka pravidla používaná primárne pre zabezpečenie kvality služieb (Quality of Service).
- **Celková dĺžka** obsahuje celkovú veľkosť paketu, do ktorej je zarátaná aj veľkosť hlavičky.
- **Identifikácia** je položka potrebná pri zostavovaní správy z fragmentovaných paketov.
- **Príznaky** rozhodujú o tom, či paket môže byť pri prenose po sieti fragmentovaný (don't fragment flag) alebo indikujú, že správa je zložená z viacerých paketov (more fragments flag).

- **Posun fragmentu** ide o množstvo už odoslaných bajtov fragmentovanej správy.
- **TTL** je počet smerovačov, cez ktoré paket môže pri smerovaní prejsť než sa zahodí. Ukážka smerovania je zobrazená na obrázku 2.1. Úlohou tejto položky je zamedziť vzniku smerovacích slučiek.
- **Protokol** značí typ transportného protokolu.
- **Kontrolný súčet** slúži pre overenie chýb v hlavičke, ktoré mohli vzniknúť pri prenose sieťou.

Obr. 2.2: Hlavička paketu IPv4 protokolu

2.2.2 Protokol IPv6

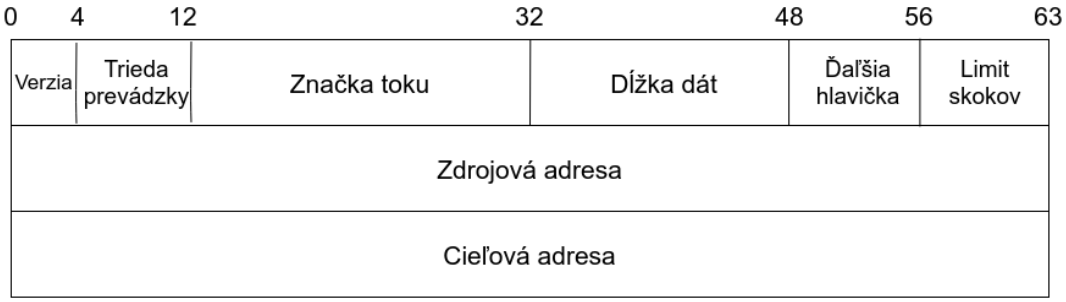
Verzia IPv6 pracuje s 128-bitovými adresami, ktoré sú zložené z 8 hexadecimálnych čísiel s možnými hodnotami od 0000 do FFFF oddelených dvojbodkou, napríklad fec0:0000:0000:0-00a:f563:5add:6fc4:152e/64 [22]. V prípade, že adresa obsahuje nulové hodnoty je možné zápis adresy podľa určitých pravidiel skrátiť [14]. Postupné skracovanie zápisu IPv6 adresy pomocou skracovacích pravidiel sa nachádza v číslovanom zozname nižšie. Adresa sa rovnako ako u verzie IPv4 skladá z identifikátora siete a rozhrania, ktoré sú určené dĺžkou prefixu. Dĺžka prefixu má u verzie IPv6 maximálnu hodnotu 128.

Postupné skracovanie zápisu IPv6 adresy:

1. Nahradenie najdlhšieho reťazca za sebou idúcich núl dvojbodkou:
   2041:0000:00FF:0000:0000:0000:875B:000A → 2041:0000:00FF::875B:000A
2. Nahradenie hexadecimálnych čísiel 0000 jednou nulou:
   2041:0000:00FF::875B:000A → 2041:0:00FF::875B:000A
3. Odstránenie úvodných zbytočných núl jednotlivých hexadecimálnych čísiel:
   2041:0:00FF::875B:000A → 2041:0:FF::875B:A

Hlavička IPv6 paketu je zobrazená na obrázku 2.3. Čísla nad hlavičkou značia veľkosti položiek v bitoch iba v prvom riadku uvedenej hlavičky. Položky hlavičky zdrojová a cieľová IP adresa majú veľkosť 128 bitov. Jednotlivé položky hlavičky (okrem adries) sú stručne opísané v nasledujúcom zozname:

- **Verzia** značí verziu hlavičky paketu (IPv4 alebo IPv6).
- **Trieda prevádzky** je premenovaná položka typ služby z hlavičky paketu IPv4 protokolu.
- **Značka toku** je položka identifikujúca jeden sieťový tok. Sieťový tok je postupnosť paketov s rovnakými vlastnosťami, ktoré prechádzajú jedným bodom siete za určitý časový interval [2].
- **Dĺžka dát** obsahuje veľkosť dát prenášaných paketom.
- **Ďalšia hlavička** je položka, ktorá špecifikuje typ ďalšej hlavičky (typ transportného protokolu alebo typ rozšírenej hlavičky).
- **Limit skokov** je premenovaná položka TTL z hlavičky paketu IPv4 protokolu.



Obr. 2.3: Hlavička paketu IPv6 protokolu

2.3 Transportná vrstva

Transportná vrstva je implementovaná iba v koncových uzloch siete. Typickou črtou koncových uzlov je, že na nich bežia rôzne aplikácie a služby. Úlohou transportnej vrstvy je vytvárať logické spojenia medzi po sieti komunikujúcimi aplikáciami a službami koncových uzlov. Aplikácie si môžu vybrať, či pre prenos dát použijú rýchlejší ale nespoľahlivý prenos dát UDP (User Datagram Protocol), alebo spoľahlivý spojovaný prenos pomocou TCP (Transmission Control Protocol). Transportná vrstva smeruje dáta konkrétnym aplikáciám, ktoré o nich požiadali. Pre identifikáciu komunikujúcich aplikácií sa používajú čísla portov. Čísla portov môžu mať hodnoty v rozsahu od 0 až po 65535. Organizácia IANA (Internet Assigned Numbers Authority) zodpovedá za priraďovanie čísiel portov špecifickým aplikáciám [7].

Rozdelenie portov podľa organizácie IANA:

- **Známe porty:** 0–1023
- **Registrované porty:** 1024–49151
- **Privátne, nepoužívané porty:** 49152–65535

Pre účely práce rozdeľujeme hodnotu portu alebo rozsah hodnôt portov do tried uvedených v nasledujúcom zozname. Toto rozdelenie bolo navrhnuté autormi článku o nástroji ClassBench [17].

- **Ľubovoľná hodnota:** (Wildcard-WC) Ľubovoľná hodnota portu. Wildcard je často označovaný pod reťazcom any alebo pod znakom hviezdička.
- **Presná hodnota:** (Exact match-EM) Presná hodnota portu, napríklad 80.
- **Ľubovoľný rozsah:** (Arbitrary range-AR) Ľubovoľný rozsah hodnôt portov, napríklad 3000–4000.
- **Presný rozsah 0–1023:** (Low ports-LO)
- **Presný rozsah 1024–65535:** (High ports-HI)

2.4 Aplikačná vrstva

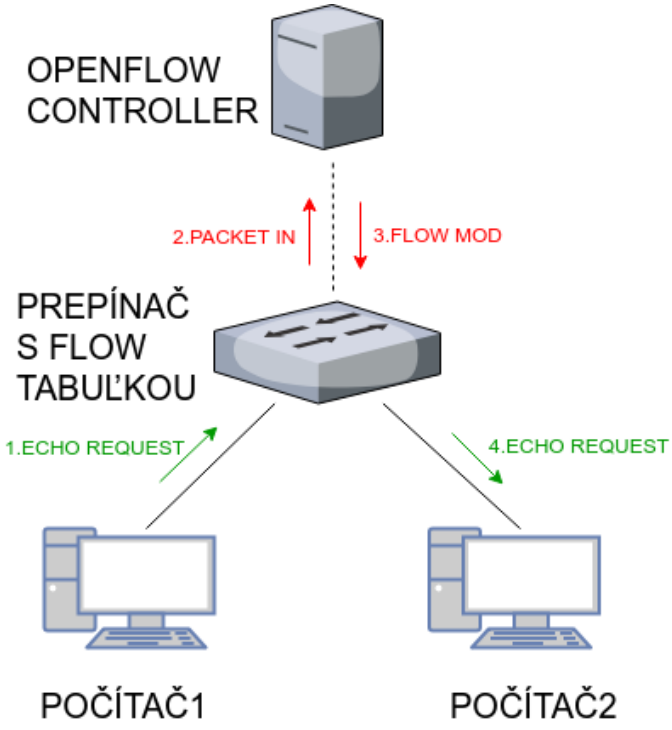
Aplikačná vrstva je najvyššia vrstva TCP/IP modelu a je zložená z aplikácií využívajúcich sieťový prenos konkrétnych dát. Každá aplikácia má pre spracovanie požiadaviek a komunikáciu určený svoj vlastný protokol. Napríklad pri zadaní adresy webového servera do prehliadača sa zavolá HTTP/HTTPS protokol, aby spracoval požiadavku. Ďalšími známymi protokolmi vrstvy sú: FTP, WWW, SSH, SMTP, DNS a DHCP. Každý aplikačný protokol pre prenos dát využíva jednu alebo obidve služby transportnej vrtvy (TCP, UDP) a je identifikovateľný číslom portu. Sieťové spojenie aplikácie je potom jednoznačne dané IP adresou, transportným protokolom a číslom portu.

2.5 Protokol OpenFlow

OpenFlow je štandardizovaný protokol umožňujúci na prepínačoch a smerovačoch softvérovo rozhodovať o tom, čo spraviť s prijatými paketmi na jeho vstupných sieťových rozhraniach [18]. Pri rozhodovaní mu pomáhajú záznamy vo Flow tabuľkách. V zmienenej tabuľke sa na základe položiek uvedených v hlavičke prijatého paketu nájde záznam, ktorý rozhodne o jeho ďalšom spracovaní. Napríklad záznam môže rozhodnúť, že paket bude zahodený, preposlaný na jedno rozhranie, preposlaný na viacero rozhraní, alebo budú zmenené položky v jeho hlavičke a podobne. V prípade, že sa vo Flow tabuľke nájde viacero záznamov pre prijatý paket, vyberie sa záznam s vyššou prioritou. Ak Flow tabuľka nemá pre prijatý paket žiaden záznam, kontaktuje sa OpenFlow controller (Packet-IN správa). Controller potom môže jednorazovo určiť, čo spraviť s paketom (Packet-OUT správa) alebo do Flow tabuľky zariadenia pridá nový záznam určujúci, čo spraviť s týmto paketom a budúcimi paketmi s podobnými hlavičkami (Flow-MOD správa). Vo Flow tabuľkách prebieha určitá dynamika. Flow-MOD správy zapĺňajú tabuľky novými záznamami a staré záznamy sú po určitej dobe nečinnosti z tabuliek vymázávané.

Popis fungovania protokolu OpenFlow uvedeného na obrázku 2.4:

1. POČÍTAČ1 chce overiť dostupnosť POČÍTAČA2.
2. PREPÍNAČ nemá vo svojej Flow tabuľke uvedený žiaden záznam pre hlavičku paketu od POČÍTAČA1. Nevie, ako má naložiť s danou požiadavkou. Na CONTROLLER preto zasiela Packet-IN správu.
3. CONTROLLER predstavuje naprogramovanú logiku, ktorá rozhoduje, čo spraviť s prijatými paketmi na rozhraní PREPÍNAČA. Konkrétne v tomto prípade zasiela na PREPÍNAČ Flow-MOD správu, ktorá pridá do Flow tabuľky zariadenia nový záznam. Záznam napríklad určí, že všetky pakety so stanovenou hlavičkou od POČÍTAČA1 budú zaslané na POČÍTAČ2.
4. Zaslanie echo request správy na POČÍTAČ2.



Obr. 2.4: Fungovanie protokolu OpenFlow

Kapitola 3

Klasifikácia paketov a generovanie umelých sád klasifikačných pravidiel

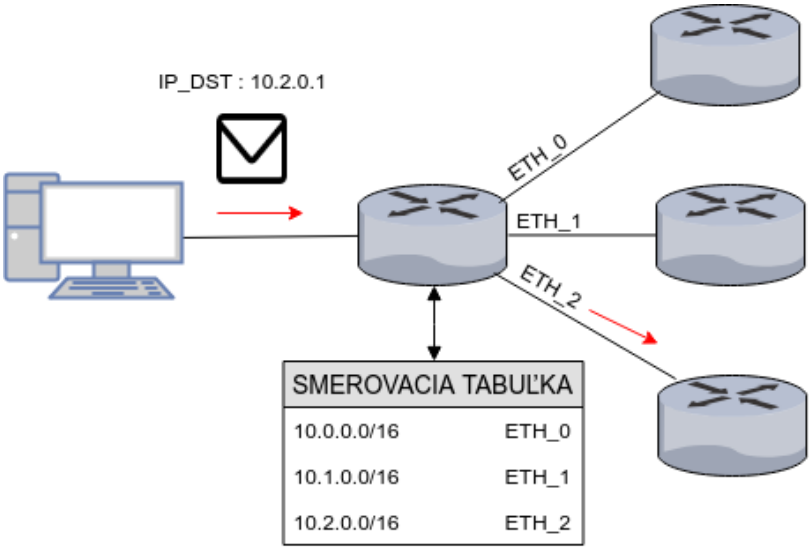
Tretia kapitola je rozdelená na 2 podkapitoly. V prvej podkapitole 3.1 sú vysvetlené nasledujúce pojmy: klasifikácia paketov, klasifikačné pravidlo a klasifikačný algoritmus. Taktiež sú tu rozpísané typické využitia klasifikácie paketov. V druhej podkapitole 3.2 sú rozoberané dôvody a spôsoby generovania umelých sád klasifikačných pravidiel pomocou voľne dostupných nástrojov ClassBench [17] a ClassBench-ng [9].

3.1 Klasifikácia paketov

Klasifikácia paketov je proces triedenia paketov prichádzajúcich na rozhranie sieťového zariadenia. Klasifikácia paketov má vo svete sietí obrovský význam a má množstvo využití. Bez neho by smerovače neboli schopné rozhodnúť, na ktoré rozhranie ďalej smerovať pakety, alebo firewally nevedeli, ktoré pakety majú byť odfiltrované na sieťovej prevádzke. Zariadenia, ktoré prevádzajú klasifikáciu paketov, majú v sebe uloženú databázu klasifikačných pravidiel usporiadanú podľa priority. Klasifikačný algoritmus má za úlohu, na základe položiek uložených v hlavičke prijatého paketu pre neho nájsť odpovedajúce pravidlo z databáze zariadenia. V prípade, že algoritmus nájde viac než jedno pravidlo, vyberie sa pravidlo s najvyššou prioritou. Zariadenie potom podľa nájdeného pravidla vie, čo s paketom ďalej spraviť. Napríklad bude vedieť, na ktoré rozhranie má byť smerovaný paket alebo rozhodnúť, že paket má byť zahodený. Klasifikácia paketov sa využíva napríklad pri: smerovaní paketov, filtrovaní paketov, správe siete alebo monitorovaní siete.

3.1.1 Smerovanie paketov

Smerovanie paketov je ilustrované na obrázku 3.1. Ide o formu klasifikácie paketov vykonávanú na základe IP adries. Na vstupné rozhranie smerovača prichádza paket. Klasifikátor zariadenia prechádza celú smerovaciu tabuľku aby našiel, čo najdlhší zhodný prefix uvedený v tabuľke s IP adresou cieľa uvedenou v hlavičke paketu. Pre nájdený prefix potom z tabuľky bude vedieť, na ktoré rozhranie má paket ďalej poslať.



Obr. 3.1: Klasifikácia paketu s cieľovou IP adresou 10.2.0.1

3.1.2 Filtrovanie paketov

Filtrovanie paketov je uskutočnené na základe filtrovacích pravidiel uložených v databáze zariadenia vykonávajuceho filtrovanie (napríklad firewall). V prípade, ak sledované položky hlavičky paketu na rozhraní splňujú všetky podmienky pravidla, potom klasifikačný algoritmus označí pravidlo za odpovedajúce paketu a vykoná s paketom akciu (zahoď alebo povoľ) uvedenú v pravidle. Najčastejšie sledované položky hlavičky paketu sú: transportný protokol, zdrojová IP adresa, cieľová IP adresa, číslo zdrojového portu a číslo cieľového portu. Položky hlavičky vypísané v predchádzajúcej vete sú označované aj ako štandardná IP pätica. Príkladom štandardnej IP pätice vo význame prvkov (transportný protokol, zdrojová IP adresa, zdrojový port, cieľová IP adresa, cieľový port) je napríklad pätica (tcp, 147.229.2.218/31, any, 147.229.131.243/32, 80).

Typy podmienok nachádzajúcich sa vo filtrovacích pravidlách [19]:

- **Rozsah hodnôt:** Určený rozsah hodnôt, ktoré vyhovujú pravidlu. Tento typ podmienky je typicky využitý pre definovanie rozsahov portov.
- **Prefix:** Určený začiatok slova, ktorým musí položka hlavičky začínať aby splňovala danú podmienku. Tento typ podmienky je typicky využitý pre určovanie IP adries, kde prefix odpovedá adrese siete. Ostatné adresy patriace do tejto siete potom tiež vyhovujú tomuto pravidlu.
- **Hodnota:** Položka hlavičky musí mať presne definovanú hodnotu.
- **Ľubovoľná hodnota:** Položka hlavičky môže mať ľubovoľnú hodnotu. Ľubovoľná hodnota je ináč označovaná aj ako wildcard.

3.1.3 Klasifikačné algoritmy

Zariadenie pre klasifikáciu paketov môže obsahovať tisícky pravidiel, ktorými klasifikačné algoritmy musia prechádzať, aby pre hlavičku paketu našli pravidlo. Od algoritmov je vždy vyžadovaná, čo najväčšia rýchlosť vyhľadania pravidla pri, čo najmenšej pamäťovej spotrebe. Neexistuje ideálny klasifikačný algoritmus, každý má vždy svoje výhody a nevýhody. V nasledujúcom zozname sú, pre predstavu ako klasifikačné algoritmu fungujú, uvedené popisy 3 jednoduchých klasifikačných algoritmov [19]:

- **Lineárne prehľadávanie:** Najjednoduchším algoritmom je lineárne prehľadávanie. Hlavička paketu sa porovnáva s každým pravidlom pokiaľ sa nenájde pravidlo, ktoré vyhovuje všetkým sledovaným položkám. Tento algoritmus je pamäťovo efektívny, ale čas výpočtu klasifikácie paketu rastie lineárne s množstvom pravidiel.
- **Vyhľadávanie po slovách:** Opačným extrémom oproti lineárnemu vyhľadávaniu je algoritmus, ktorý z položiek hlavičiek paketu vytvorí jedno slovo a použije ho ako adresu do pamäti. Pre vyhľadanie pravidla potom stačí jeden prístup do pamäte, avšak pamäťová náročnosť takéhoto algoritmu je obrovská. Pre vyhľadávanie pravidiel podľa štandardnej IP pätice, by pamäť musela obsahovať 2^{104} položiek.
- **TCAM:** Často využívaným riešením pre klasifikáciu paketov v komerčných zariadeniach je využívanie asociatívnych pamätí. Vyhľadávanie pravidiel v takejto pamäti je spracovávané paralelne, a teda veľmi rýchle. Najviac využívanou variantou asociatívnych pamätí je ternárna varianta (TCAM), ktorá okrem núl a jedničiek pracuje ešte s tretím stavom `do not care`. Tomuto stavu je potom pri vyhľadávaní pravidla odpovedajúcemu hlavičke paketu jedno, či bit v položke hlavičky paketu je jedna alebo nula. Nevýhodou asociatívnych pamätí je, že majú malú kapacitu, vysoký príkon a sú drahé.

3.2 Generovanie umelých sád klasifikačných pravidiel

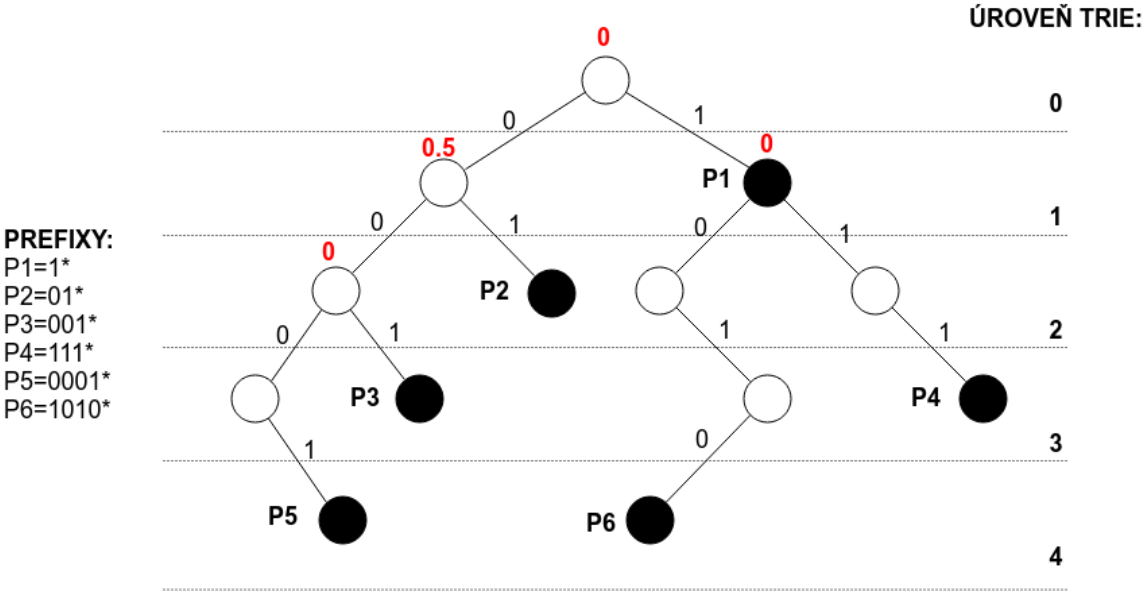
Dôvod, prečo generovať umelé sady filtrovacích pravidiel, je možnosť testovať výkonnosť klasifikačných algoritmov. Pri testovaní na rozhranie zariadenia vykonávajuceho klasifikáciu prichádzajú pakety a testovaný algoritmus musí z vygenerovanej sady pravidiel nájsť jedno pravidlo, ktoré odpovedá sledovaným položkám hlavičky paketu. Ako bolo spomínané v sekcii 3.1.3, výkonnosť klasifikačných algoritmov závisí na konkrétnych položkách pravidiel. Z tohto tvrdenia potom vyplýva, že najlepšou sadou pre testovanie je reálna sada, ktorá sa skutočne nachádza v databáze zariadenie, a nie vygenerovaná umelá sada pravidiel. Problémom však je nedostupnosť reálnych sád. Reálne sady sú uložené vo firewalloch a smerovačoch organizácií, a teda z hľadiska bezpečnosti sieti organizácie nemôžu byť zverejnené. Preto sa pre testovanie algoritmov používajú umelé sady.

Programy ClassBench [17] a jeho vylepšená verzia ClassBench-ng [9] poskytujú generovanie umelých pravidlových sád. Generovanie pravidiel obidvoch nástrojov je riadené na základe parametrového súboru obsahujúceho štatistiky a rozdelenia pravdepodobnosti položiek pravidiel reálnej sady. Takýto spôsob generovania uchováva anonymitu citlivých údajov uvedených v pravidlách sady (IP a MAC adresy zariadení) a produkuje umelú sadu, ktorá modeluje štruktúru reálnej sady. Krátky popis nástrojov a ich formáty parametrových súborov sú popísané v nasledujúcich sekciách 3.2.1 a 3.2.2.

3.2.1 Nástroj ClassBench

ClassBench predstavuje sadu nástrojov pre testovanie klasifikačných algoritmov. Okrem generovania umelých sád pravidiel umožňuje aj generovanie hlavičiek paketov, ktoré pri testovaní algoritmu vhodne otestujú vygenerovanú sadu. Položky vygenerovaných pravidiel sa skladajú z prvkov štandardnej IP pätice. Príklad parametrového súboru nástroja ClassBench sa nachádza v prílohe B.

Pre výpočet niektorých parametrov je potrebné skúmať binárne prefixové stromy zostrojené zo zdrojových alebo cieľových prefixov pravidiel reálnej sady. Prefixový binárny strom býva označovaný aj pod pojmom trie. Príklad trie je ukázaný na obrázku 3.2. Trie na uvedenom obrázku je zostrojený z prefixov P1 až P6, vypísaných vedľa grafického zobrazenia stromu, v zozname vľavo. Zo zostrojeného trie je pre parametrový súbor potrebné získať nasledujúce 2 rozdelenia: rozdelenie pravdepodobnosti vetvenia uzlov stromu na jednotlivých úrovňach stromu a rozdelenie priemerov skew hodnôt na jednotlivých úrovňach stromu. Úroveň stromu je hodnota v rozsahu 0 až 32 a určuje množinu skúmaných prefixov stromu s rovnakými dĺžkami. Úroveň stromu 0 sa nazýva koreň stromu a môže obsahovať prázdny prefix (ekvivalent wildcardu).



Obr. 3.2: Jednoduchý binárny prefixový strom

Rozdelenie pravdepodobnosti vetvenia uzlov stromu udáva, aká je pravdepodobnosť, že uzol na jednotlivých úrovňach stromu má jedného alebo dvoch potomkov. Rozdelenie priemerov skew hodnôt udáva priemer skew hodnôt na jednotlivých úrovňach stromu. Na obrázku 3.2 sú vypočítané skew hodnoty vykreslené nad uzlami stromu. Skew hodnoty sa počítajú iba pre uzly s 2 potomkami na základe počtu prefixov v podstromoch uzla podľa vzorca:

$$skew = 1 - \frac{mensi\_pocet\_prefixov}{vacsi\_pocet\_prefixov}$$

Posledný údaj z binárneho prefixového stromu, ktorý je pre parametrový súbor dôležitý, je parameter prahu zanorenia prefixov, takzvaný prefix nesting treeshold. Prefix nesting treeshold udáva maximálny počet prefixov na ľubovoľnej ceste od koreňa stromu až ku jeho listom. Pre lepšie pochopenie výpočtov parametrov z binárneho prefixového stromu sú uvedené príklady výpočtov: pravdepodobnosti vetvenia uzlov na úrovni stromu 2, priemeru skew hodnôt na úrovni stromu 0 a parametru prahu zanorenia prefixov. Hodnoty sú počítané pre strom zobrazený na obrázku 3.2.

Príklady výpočtov potrebných pre získanie parametrov z trie:

1. Pravdepodobnosť vetvenia uzlov na úrovni stromu 2

   (a) Pravdepodobnosť, že uzol na úrovni stromu 2 má 1 potomka:

   $$p_1 = \frac{2}{3} = 0.66$$

   (b) Pravdepodobnosť, že uzol na úrovni stromu 2 má 2 potomkov:

   $$p_2 = \frac{1}{3} = 0.33$$

2. Priemer skew hodnôt na úrovni stromu 0:

   $$skew = 1 - \frac{mensi\_pocet\_prefixov}{vacsi\_pocet\_prefixov} = 1 - \frac{3}{3} = 0$$

3. Prah zanorenia prefixov: Hodnotu parametru nie je potrebné počítať. Dá sa ju odčítať z grafického zobrazenia stromu. Z obrázku je vidno, že maximálny počet prefixov na ľubovoľnej ceste od koreňa stromu až ku jeho listom je 2, a to pri ceste cez prefixy P1 a P6 alebo pri ceste cez prefixy P1 a P4.

Formát parametrového súboru nástroja ClassBench sa skladá z týchto parametrov:

- **scale** Celkový počet pravidiel v reálnej sade.
- **prots** Rozdelenie pravdepodobnosti protokolov v reálnej sade. Protokoly sú číslované podľa organizácie IANA [6]. Napríklad číslo 6 hovorí, že ide o TCP protokol. Číslo 0 označuje ľubovoľný protokol (wildcard). Pre každý jedinečný protokol je ďalej uvedené rozdelenie pravdepodobnosti pravidiel do PPC (Port Pair Class). PPC je usporiadaná dvojica: (trieda zdrojového portu a trieda cieľového portu pravidla). Rozdelenie portov do tried je popísané v kapitole 2.3. Každé pravidlo teda môžme zaradiť do jednej z 25 možných kombinácií dvojíc tried portov označovaných ako PPC. Napríklad pravidlo s hodnotou zdrojového portu 80 a cieľového portu 0–1023, zaradíme do PPC EM-LO.
- **flags** Rozdelenie pravdepodobnosti príznakov pre určité protokoly v reálnej sade. Príznaky sú označované 4-miestnym hexadecimálnym číslom.
- **spar** Rozdelenie pravdepodobnosti ľubovoľných rozsahov zdrojových portov v reálnej sade.
- **spem** Rozdelenie pravdepodobnosti presných hodnôt zdrojových portov v reálnej sade.

- **dpar** Rozdelenie pravdepodobnosti ľubovoľných rozsahov cieľových portov v reálnej sade.
- **dpem** Rozdelenie pravdepodobnosti presných hodnôt cieľových portov v reálnej sade.
- **rozdelenie dĺžok prefixov** Rozdelenie pravdepodobnosti celkových dĺžok prefixov pravidiel s rovnakou PPC v reálnej sade. Keďže existuje 25 PPC, je potom pre každú PPC v parametrovom súbore uvedené vlastné rozdelenie pravdepodobnosti celkových dĺžok prefixov. Pod pojmom celková dĺžka prefixov rozumieme súčet dĺžky prefixu zdrojovej IP adresy a dĺžky prefixu cieľovej IP adresy. Pre každú celkovú dĺžku prefixu je v parametri ďalej uvedené rozdelenie pravdepodobnosti dĺžok zdrojových prefixov, z ktorých sa celková dĺžka prefixu počítala.
- **snest** Parameter prefix nesting treeshold binárného prefixového stromu zostrojeného zo zdrojových prefixov reálnej sady.
- **sskew** Rozdelenie pravdepodobnosti vetvenia uzlov a rozdelenie priemerov skew hodnôt na jednotlivých úrovňach binárneho prefixového stromu zostrojeného zo zdrojových prefixov reálnej sady.
- **dnest** Parameter prefix nesting treeshold binárného prefixového stromu zostrojeného z cieľových prefixov reálnej sady.
- **dskew** Rozdelenie pravdepodobnosti vetvenia uzlov a rozdelenie priemerov skew hodnôt na jednotlivých úrovňach binárneho prefixového stromu zostrojeného z cieľových prefixov reálnej sady.
- **pcorr** Pravdepodobnosti pre jednotlivé úrovne dĺžok prefixov 1 až 32 udávajúce šancu, že binárne zápisy zdrojových a cieľových prefixov pravidiel reálnej sady sú na určitej úrovni rovnaké. Porovnávanie pravidla končí, keď binárny zápis zdrojového a cieľového prefixu prestane byť od určitej úrovni rovnaký. Binárne zápisy zdrojových a cieľových prefixov sú porovnávané až po veľkosť menšieho z prefixov.

3.2.2 Nástroj ClassBench-ng

ClassBench-ng rozširuje pôvodný nástroj o podporu IPv6 adries, rozširuje položky generovaných pravidiel o položky protokolu OpenFlow a obsahuje analýzator reálnych sád pravidiel. Analyzátor je časť programu, ktorá prevádza výpočty štatistík a rozdelení pravdepodobností reálnej sady a ukladá ich do parametrového súboru. Príklad parametrového súbora nástroja ClassBench-ng sa nachádza v prílohe C. Formát parametrového súboru nástroja ClassBench-ng sa skladá zo všetkých parametrov formátu nástroja ClassBench. Formát je však naviac rozšírený o tieto parametre týkajúce sa položiek OpenFlow protokolu:

- **in_port** Pre každú unikátnu hodnotu vstupného portu nachádzajúcu sa v reálnej OpenFlow sade je vypísaný počet, koľko krát sa daná hodnota v sade vyskytuje.
- **eth_type** Pre každý unikátny typ položky pravidla EtherType nachádzajúci sa v reálnej OpenFlow sade je vypísaný počet, koľko krát sa daný typ v sade vyskytuje.
- **dl_src** Pre každú unikátnu hodnotu zloženú z prvých 6 hexadecimálnych čísiel zdrojovej MAC adresy nachádzajúcej sa v reálnej OpenFlow sade je vypísaný počet, koľko

krát sa daná hodnota v sade vyskytuje. Ako je uvedené v podkapitole 2.1, prvých 6 hexadecimálnych čísiel MAC adresy identifikuje iba výrobcu zariadenia a neidentifikuje konkrétne zariadenie. Tým pádom je zachovaná anonymita zariadení.

- **dl_dst** Pre každú unikátnu hodnotu zloženú z prvých 6 hexadecimálnych čísiel cieľovej MAC adresy nachádzajúcej sa v reálnej OpenFlow sade je vypísaný počet, koľko krát sa daná hodnota v sade vyskytuje.
- **unique_vlan_ids_count** Počet unikátnych identifikátorov VLAN (Virtual Local Area Network).
- **empty_rules_count** Počet pravidiel reálnej sady bez špecifikovania položiek OpenFlow protokolu.
- **rule_type** Rozdelenie reálnej OpenFlow sady do skupín pravidiel s rovnakými špecifikovanými OpenFlow položkami. Pre každú skupinu je vypísaný zoznam názvov špecifikovaných OpenFlow položiek a uvedený počet pravidiel v skupine.

Kapitola 4

Návrh univerzálneho analyzátora reálnych sád klasifikačných pravidiel

Jedným z nedostatkov nástrojov ClassBench [17] a ClassBench-ng [9] je ich slabá podpora pre tvorbu parametrových súborov z reálnych sád pravidiel. V publikovanej verzii nástroja ClassBench nie je analyzátor reálnych sád dostupný vôbec. ClassBench-ng obsahuje analyzátor reálnej sady, no je funkčný iba pre určitý formát reálnej sady. Úlohou práce je implementovať analyzátor, ktorý je schopný vytvoriť parametrový súbor z ľubovoľného formátu reálnej sady, ktorej pravidlá sú zložené z prvkov štandardnej IP pätice. Pojem štandardná IP pätica je vysvetlený v sekcii 3.1.2.

Piata kapitola sa zaoberá návrhom cieľového nástroja práce pomenovaného ako analyzátor reálnej sady klasifikačných pravidiel. Analyzátor je konzolová aplikácia, ktorej vstupom je reálna sada klasifikačných pravidiel a jej výstupom je súbor s parametrami potrebnými pre generovanie umelej sady pravidiel v nástrojoch ClassBench a ClassBench-ng. Z ktorých parametrov sa skladajú parametrové súbory a ako sa parametre počítajú je uvedené pre nástroj ClassBench v sekcii 3.2.1 a pre nástroj ClassBench-ng v sekcii 3.2.2. Cieľom navrhovaného analyzátora je vedieť vytvoriť parametrový súbor formátu ClassBench. Aby analyzátor vedel vytvoriť parametrový súbor formátu ClassBench zo všetkých položiek pravidiel vo vstupných sadách, analyzátoru stačí sledovať iba položky pravidiel, ktoré sú súčasťou štandardnej IP pätice. Položky pravidiel OpenFlow protokolu, ktoré by boli potrebné pre tvorbu parametrového súboru formátu ClassBench-ng, sú pre návrh tohto analyzátora nezaujímavé.

Najdôležitejšia vlastnosť, ktorá je od navrhovaného analyzátora požadovaná, je univerzálnosť. Univerzálnosť pre analyzátor znamená schopnosť vytvoriť parametrový súbor zo všetkých existujúcich formátov reálnych sád pravidiel. Nástroj ClassBench-ng obsahuje analyzátor reálnych sád, no nie je univerzálny. Analyzátor ClassBench-ng vie pracovať iba s reálnou sadou pravidiel vo formáte pravidiel používanom v nástroji ovs-ofctl.

4.1 Známe formáty klasifikačných pravidiel reálnych sád

Pre účel návrhu riešenia univerzálnosti analyzátora táto podkapitola rozoberá formáty klasifikačných pravidiel používaných v nástrojoch iptables, IPFW a ovs-ofctl. V úvode kapitoly bolo spomenuté, že aby analyzátor vedel vytvoriť parametrový súbor formátu ClassBench,

stačí mu sledovať položky pravidiel, ktoré sú súčasťou štandardnej IP pätice. Tieto položky sú v kapitole ďalej označované ako užitočné položky pravidiel. Nasledujúce rozbory formátov klasifikačných pravidiel sa práve primárne zameriavajú na zápis týchto užitočných položiek pravidiel v jednotlivých formátoch nástrojov.

4.1.1 Formát nástroja iptables

Iptables je nástroj pre správu sietí na zariadeniach s operačným systémom s jadrom Linuxu. Nástroj ponúka tieto funkcie: softvérový firewall umožňujúci filtrovanie paketov, NAT (Network Address Translation) umožňujúci preklady IP adries a funkciu mangle pre modifikáciu položiek typ služby alebo TTL v hlavičkách paketov. Jednotlivé položky hlavičiek paketov sú popísané v podkapitole 2.2. Pre každú z vymenovaných funkcií má nástroj v systéme uloženú vlastnú tabuľku s pravidlami definujúcimi chovanie funkcie. Napríklad pre fungovanie softvérového firewallu sa v systéme nachádza tabuľka s filtrovacími pravidlami, ktoré rozhodujú o tom, či má byť paket na rozhraní zariadenia zahodený, alebo ho je možné smerovať ďalej. Obdobne pre fungovanie NAT sa v systéme nachádza tabuľka s pravidlami pre preklad IP adries v hlavičkách paketov na rozhraní zariadenia. Pre pochopenie formátu pravidiel nástroja iptables sú ďalej skúmané časti príkazu nástroja pre manipulovanie s pravidlami v tabuľkách. Formát príkazu nástroja iptables pre manipulovanie s pravidlami v tabuľkách nástroja iptables [5]:

iptables [-t TABLE] COMMAND CHAIN [INDEX] PARAMETER -j TARGET

Popis jednotlivých častí uvedeného formátu príkazu sa nachádza v nasledujúcom zozname:

- **TABLE** je časť príkazu uvedená prepínačom `-t` určujúca, s ktorou tabuľkou nástroja sa manipuluje. Jej možné hodnoty sú: `filter` pre manipuláciu s filtrovaciou tabuľkou, `nat` pre manipuláciu s NAT prekladovou tabuľkou a `mangle` pre manipuláciu s tabuľkou mangle pravidiel. Pri vynechaní tejto časti príkazu sa manipuluje s filtrovaciou tabuľkou pravidiel.

- **COMMAND** určuje, o akú manipuláciu s tabuľkou ide. Do tabuľky je možné pridať nové pravidlá, či modifikovať alebo odstrániť už uložené pravidlá. Taktiež je možné vypísať zoznam všetkých pravidiel v tabuľke alebo len zoznam pravidiel v určitom reťazci (`chain`) tabuľky.

- **CHAIN** rozhoduje, s ktorým reťazcom tabuľky sa manipuluje. Reťazec je určitá podmnožina pravidiel v tabuľke. Každá tabuľka vo východzom nastavení obsahuje 3 vstavané reťazce: INPUT, OUTPUT a FORWARD. Ako ich anglické názvy napovedajú, pokiaľ paket prichádza na rozhranie zariadenia, aplikujú sa na neho pravidlá z reťazca INPUT. Pravidlá z reťazca OUTPUT sa aplikujú pre odchadzajúce pakety z rozhrania zariadenia. A nakoniec pravidlá z reťazca FORWARD sa aplikujú, keď zariadenie funguje ako smerovač a preposiela pakety medzi sieťami.

- **INDEX** určuje, s ktorým konkrétnym pravidlom z reťazca sa manipuluje. Pravidlá v reťazci sú identifikované indexom, teda číselným poradím pravidla v reťazci. Ide o voliteľnú časť príkazu, pretože nie pre každú manipuláciu s reťazcom je potrebné uvádzať číslo indexu pravidla. Napríklad pri vkladaní nového pravidla do reťazca s prepínačom `-A` sa pravidlo jednoducho uloží na ďalší voľný index. Index pravidiel v reťazci rozhoduje o priorite pravidla v prípade, ak paket na rozhraní vyhovuje

viacerým pravidlám reťazca. Menší index má väčšiu prioritu. Ak paket nevyhovuje žiadnemu pravidlu v reťazci, aplikuje sa na neho východzie nastavenie reťazca.

- **PARAMETER** je časť príkazu definujúca užitočné položky pravidla. V nasledujúcom podzozname sa nachádzajú zkrátené a v zátvorke dlhé tvary prepínačov v príkaze uvádzajúcich užitočné položky pravidiel.

  `-p (--protocol)` Prepínač uvádzajúci protokol.

  `-s (--source)` Prepínač uvádzajúci zdrojovú IP adresu.

  `-d (--destination)` Prepínač uvádzajúci cieľovú IP adresu.

  `--sport (--source-port)` Prepínač uvádzajúci zdrojový port alebo rozsah.

  `--dport (--destination-port)` Prepínač uvádzajúci cieľový port alebo rozsah.

- **TARGET** je časť príkazu uvedená prepínačom `-j` určujúca, čo sa má stať s paketom na rozhraní, ktorého hlavička vyhovuje pravidlu definovaného príkazom. Paket, ktorého hlavička vyhovuje pravidlu vo filtrovacej tabuľke, môže byť prijatý alebo zahodený.

Príklad príkazu nástroja iptables:

```
iptables -A INPUT -s 192.168.0.0/24 -p tcp --dport 25 -j DROP
```

Uvedený príkaz pridáva nové pravidlo na koniec reťazca INPUT patriaceho filtrovacej tabuľke. Definované pravidlo zahadzuje všetky pakety prichadzajúce na rozhranie zariadenia zo siete 192.168.0.0/24 s určeným TCP protokolom a cieľovým portom 25.

4.1.2 Formát nástroja IPFW

Nástroj IPFW je veľmi podobný nástroju iptables. Je však dostupný iba na operačných systémoch typu FreeBSD. IPFW poskytuje rovnaké funkcie ako iptables: softvérový firewall a NAT. Jeho formát príkazu pre manipuláciu s filtrovacími pravidlami je však trochu iný. Formát príkazu nástroja IPFW pre manipuláciu s filtrovacími pravidlami [3]:

ipfw COMMAND [INDEX] ACTION PARAMETER

Popis jednotlivých častí uvedeného formátu príkazu sa nachádza v nasledujúcom zozname:

- **COMMAND** je časť príkazu, ktorá určuje, o akú manipuláciu s filtrovacími pravidlami tabuľky ide. Jej možné hodnoty sú: `add` pre pridanie nového pravidla, `delete` pre odstránenie pravidla alebo `list` pre výpis všetkých filtrovacích pravidiel firewallu.

- **INDEX** určuje, s ktorým konkrétnym filtrovacím pravidlom sa manipuluje. Pravidlá sú rovnako ako u nástroja iptables identifikované indexom, ktorý rozhoduje o priorite v prípade, ak paket na rozhraní vyhovuje viacerým pravidlám. Ak paket nevyhovuje žiadnemu filtrovaciemu pravidlu, aplikuje sa na neho pravidlo s číslom 65535, ktoré zahadzuje pakety.

- **ACTION** je časť príkazu, ktorá určuje, čo sa má stať s paketom na rozhraní, ktorého hlavička vyhovuje pravidlu definovaného príkazom. Jej možné hodnoty sú: `allow`

pre bežné prepustenie paketu prechadzujúceho rozhraním, `reject` pre zahodenie paketu s odoslaním ICMP správy o nedostupnosti cieľa zdroju paketu, `deny` pre zahodenie paketu bez odoslania ICMP správy a `count` pre aktualizáciu paketových čítačov na rozhraní.

- **PARAMETER** je časť príkazu definujúca užitočné položky pravidla. Definovanie užitočných položiek pravidiel má v nástroji IPFW špecifický formát. Definícia začína sieťovým protokolom. Nasleduje slovo `from`, zdrojová IP adresa a zdrojový port alebo rozsah portov. Definícia končí slovom `to`, cieľovou IP adresou a cieľovým portom alebo rozsahom portov. Pri definícii nie je nutné uviesť všetky užitočné položky. Ak sa položka neuvedie, pravidlo má wildcard hodnotu položky.

Príklad príkazu nástroja IPFW:

```
ipfw add deny tcp from 192.168.0.0/24 to 25
```

Uvedený príkaz pridáva nové filtrovacie pravidlo do firewallu nástroja IPFW. Definované pravidlo zahadzuje všetky pakety prichadzajúce na rozhranie zariadenia zo siete 192.168.0.0/24 s určeným TCP protokolom a cieľovým portom 25.

4.1.3 Formát nástroja ovs-ofctl

Ovs-ofctl je nástroj pre administráciu prepínačov podporujúcich OpenFlow protokol. OpenFlow je protokol umožňujúci na prepínačoch softvérovo rozhodovať o tom, čo spraviť s prijatými paketmi na jeho vstupných sieťových rozhraniach. Fungovanie protokolu OpenFlow je detailnejšie opísané v podkapitole 2.5. Formát príkazu nástroja ovs-ofctl je príliš rozsiahly, poskytuje príliš mnoho možností a nie je rozoberaný ako u predchadzujúcich dvoch nástrojov [12]. V nasledujúcom zozname sú teda uvedené iba kľúčové slová uvádzajúce užitočné položky pravidiel v ovs-ofctl formáte:

- **`nw_proto`** Kľúčové slovo uvádzajúce protokol. Za kľúčovým slovom musí nasledovať číslo protokolu definované organizáciou IANA.
- **`nw_src`** Kľúčové slovo uvádzajúce zdrojovú IP adresu.
- **`nw_dst`** Kľúčové slovo uvádzajúce cieľovú IP adresu.
- **`tp_src`** Kľúčové slovo uvádzajúce zdrojový port alebo rozsah zdrojových portov.
- **`tp_dst`** Kľúčové slovo uvádzajúce cieľový port alebo rozsah cieľových portov.

Uvádzajúce pojmy a užitočné položky sú v pravidle oddelené znakom '='. Príklad časti pravidla nástroja ovs-ofctl definujúceho všetky užitočné položky: `nw_proto=6, nw_src=192.168.0.0/24, nw_dst=192.168.1.0/24, tp_src=0:1023, tp_dst=80`.

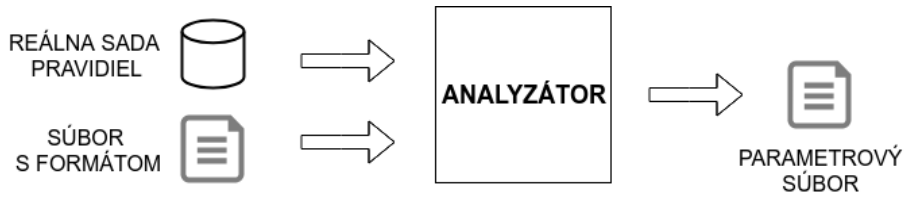
4.2 Riešenie univerzálnosti analyzátora

Rozbor formátov pravidiel z predchadzajúcej podkapitoly ukázal, že zápis pravidla sa okrem užitočných položiek skladá aj z uvádzajúcich prepínačov alebo kľúčových slov a pre analyzátor nezaujímavých častí. Uvádzajúce prepínače alebo kľúčové slová v pravidle uvádzajú užitočné položky. Napríklad formát pravidla nástroja IPFW obsahuje 2 uvádzajúce kľúčové

slová: `from` a `to`. Kľúčové slovo `from` uvádza zdrojovú IP adresu a zdrojový port alebo rozsah portov pravidla. Kľúčové slovo `to` zase uvádza cieľovú IP adresu a cieľový port alebo rozsah portov pravidla. Nezaujímavé časti pravidla sú všetky ostatné časti zápisu pravidla. Nezaujímavé časti nemajú žiadnu užitočnú informáciu pre výpočet parametrov a ani neuvádzajú užitočné položky, no v pravidlách sa nachádzajú a musí sa s nimi pri spracovaní pravidiel počítať. Nezaujímavými časťami pravidla sú vo formáte nástroja iptables napríklad: INPUT, OUTPUT, FORWARD, DROP, ACCEPT. Ďalšie zistené informácie z rozboru formátov pravidiel sú zhrnuté v nasledujúcom zozname:

- Všetky časti zápisov pravidiel sú oddelené medzerami, čiarkami alebo znakom '='.
- Užitočné položky pravidiel sú v zápisoch pravidiel voliteľné. Pri ich neuvedení sa uvažujú wildcard hodnoty užitočných položiek.
- Uvádzajúce prepínače alebo kľúčové slová sú rozdielne pre rôzne formáty pravidiel.

Na obrázku 4.1 je uvedená bloková schéma riešiacia problém univerzálnosti analyzátora. Vstupom analyzátora je reálna sada pravidiel ľubovoľného formátu a súbor s formátom reálnej sady pravidiel. Úlohou súboru s formátom sady je vhodne popísať formát jedného pravidla reálnej sady. Vhodne popísaný formát pravidlá znamená, že spustený analyzátor pri postupnom prechode cez jednotlivé pravidlá sady vie, na ktorých pozíciach sa nachádzajú jednotlivé užitočné položky pravidla, a nemá problém naplniť svoje pomocné dátové štruktúry. Po prechode cez celú sadu pravidiel sa potom z pomocných dátových štruktúr počítajú žiadané parametre a vypisujú sa na štandardný výstup alebo sa ukladajú do súboru. Spôsob výpočtu parametrov je implementačná záležitosť a je jej venovaná celá kapitola 5. V zbytku tejto kapitoly sa opisuje navrhnutý spôsob definovania súboru s formátom pravidiel reálnej sady.



Obr. 4.1: Bloková schéma analyzátora reálnej sady pravidiel

4.3 Súbor s formátom pravidiel

Formát pravidiel reálnej sady hovorí, ako by malo vyzerať jedno pravidlo v sade. Aby analyzátor vedel načítať formát pravidiel, musí sa formát nachádzať na prvom riadku súboru. Formát pravidiel sa skladá z parametrov a kľúčových slov. Parametre sú slová, ktoré reprezentujú užitočné položky pravidiel a taktiež nezaujímavé časti pravidiel. Vo formáte pravidiel je parametre možné ľahko nájsť, pretože všetky písmená ich reťazcov sú veľké. Napríklad parameter 'PROTOCOL' vo formáte reprezentuje sieťový protokol. Zoznam všetkých parametrov s vysvetlením, čo jednotlivé parametre vo formáte reprezentujú, sa nachádza nižšie. Kľúčové slová nadefinované vo formáte reprezentujú tie isté slová v pravidle. Kľúčové slová sú napríklad použité pre definovanie uvádzajúcich prepínačov alebo pojmov. Kľúčovými slovami sú napríklad reťazce: `-p`, `--source-port`, `from`, `iptables` alebo `ipfw`.

Zoznam parametrov v navrhnutom spôsobe definovania formátu pravidiel:

- **PROTOCOL** reprezentuje užitočnú položku pravidla, sieťový protokol, napríklad ICMP alebo TCP.
- **SRC_IP** reprezentuje užitočnú položku pravidla, zdrojovú IP adresu, napríklad 10.0.0.0/24 alebo 192.168.0.33.
- **SRC_PORT** reprezentuje užitočnú položku pravidla, zdrojový port alebo rozsah portov, napríklad 80 alebo 0:1023.
- **DST_IP** reprezentuje užitočnú položku pravidla, cieľovú IP adresu.
- **DST_PORT** reprezentuje užitočnú položku pravidla, cieľový port alebo rozsah portov.
- **NUMBER** reprezentuje nezaujímavú časť pravidla, dekadické číslo, ktoré nie je použité pre výpočet parametrov, napríklad číslo v zápise pravidla určujúce jeho prioritu.
- **WILDCARD** reprezentuje nezaujímavú časť pravidla, reťazec znakov. Od kľúčového slova sa parameter WILDCARD odlišuje tým, že jeho reťazec môže byť pre rôzne pravidla sady odlišný. Napríklad v jednom pravidle sady sa na mieste parametru WILDCARD môže nachádzať reťazec ACCEPT, v druhom pravidle sady sa na rovnakom mieste môže nachádzať reťazec DROP. Kľúčové slovo tiež reprezentuje reťazec znakov, no jeho reťazec musí byť rovnaký pre každé pravidlo v sade.

Poradie parametrov a kľúčových slov je vo formáte pravidiel dôležité. V procese spracovania jedného pravidla z reálnej sady sa kontroluje postupne od prvej časti pravidla až po jej poslednú časť, či každá časť pravidla sedí s jej definíciou vo formáte pravidiel. Napríklad ak definícia formátu pravidiel začína s kľúčovým slovom 'from' a za ním nasleduje parameter 'SRC_IP', očakáva sa, že prvá časť načítavaných pravidiel bude reťazec 'from' a druhá časť pravidla bude reprezentácia nejakej IP adresy. Ak poradie alebo očakávané typy časti pravidla nesedia s ich definíciou vo formáte pravidiel, pravidlo je ignorované.

Navrhnutý spôsob definovania formátu pravidiel umožňuje definovať voliteľné parametre alebo voliteľné kľúčové slová. Definícia voliteľných parametrov alebo kľúčových slov je vo formáte pravidiel možná pridaním znaku '?' za reťazcom formátu reprezentujúcim parameter alebo kľúčové slovo. Parametre alebo kľúčové slová, ktoré nekončia znakom '?', sú povinné. Napríklad ak definícia formátu pravidiel začína s kľúčovým slovom 'from' a za ním nasleduje parameter 'SRC_IP?', očakáva sa, že prvá časť načítavaných pravidiel bude reťazec 'from'. Ak to neplatí, pravidlo bude ignorované. Druhá časť načítavaných pravidiel, reprezentácia IP adresy, je voliteľná. V prípade, ak sa reprezentácia IP adresy v druhej časti načítavaných pravidiel nenachádza, pravidlo nebude ignorované a pokračuje sa spracovaním ďalších častí pravidla. V nasledujúcom zozname sa nachádzajú navrhnutým spôsobom definované formáty pravidiel nástrojov iptables, IPFW a ovs-ofctl:

- **iptables krátky formát:**
  'WILDCARD WILDCARD NUMBER? -s? SRC_IP? -d? DST_IP? -p? PROTOCOL? --sport? SRC_PORT? --dport? DST_PORT? -j WILDCARD'
  príklad pravidla s iptables krátkym formátom:
  `-A INPUT -d 192.0.0.0/3 -p udp --dport 1024:65535 -j DROP`

- **iptables dlhý formát:**
  'WILDCARD WILDCARD NUMBER? --source? SRC_IP? --destination? DST_IP? --protocol? PROTOCOL? --source-port? SRC_PORT? --destination-port? DST_PORT? -j WILDCARD'
  príklad pravidla s iptables dlhým formátom:
  `-I OUTPUT 2 --source 192.0.0.0/3 --source-port 1024:65535 -j ACCEPT`

- **IPFW:**
  'add WILDCARD NUMBER? PROTOCOL? from SRC_IP? SRC_PORT? to DST_IP? DST_PORT?'
  príklad pravidla s IPFW formátom:
  `add allow tcp from to 10.0.0.0/24 80`

- **ovs-ofctl:**
  'nw_proto?=PROTOCOL?, nw_src?=SRC_IP?, nw_dst?=DST_IP?, tp_src?=SRC_PORT?, tp_dst?=DST_PORT?'
  príklad pravidla s ovs-ofctl formátom:
  `nw_proto=17, nw_src=147.229.35.0/24, nw_dst=147.229.37.9/32`

Kapitola 5

Implementácia analyzátora reálnych sád klasifikačných pravidiel

V tejto kapitole je popísaná implementácia analyzátora reálnych sád pravidiel navrhnutého v predchadzajúcej kapitole. Analyzátor je implementovaný v programovacom jazyku Python verzie 3.6.7. Jazyk Python bol pre implementáciu analyzátora vybraný, pretože poskytuje vhodné vlastnosti pre analýzu dát. V prvej časti kapitoly sú uvedené prepínače, s akými je možné analyzátor ako konzolovú aplikáciu spustiť. Ďalšie časti sú zamerané na priblíženie algoritmu naplnenia užitočných položiek pravidiel z reálnych sád do pomocných dátových štruktúr programu, výpočet parametrového súbora a výpis parametrového súbora na štandardný výstup alebo do súboru. V priebehu práce boli implementované 2 verzie analyzátora. Porovnanie oboch verzií analyzátoru je témou záverečnej časti kapitoly.

5.1 Prepínače analyzátora

Formát príkazu pre spustenie implementovaného analyzátora v konzole operačného systému:

python3 -m filter_rule_analyzer -r <rules_file> -f <format_file>
[-o <output_file>] [-l] [-h]

Z uvedeného formátu sú v nasledujúcom zozname vypísané všetky povinné a voliteľné prepínače. Pri každej položke zoznamu je uvedený krátky a v zátvorke dlhý zápis prepínača a je vysvetlené, čo jednotlivý prepínač uvádza:

- `-m` je povinný prepínač, ktorý uvádza modul s implementovaným analyzátorom (`filter_rule_analyzer`) pre interpretér jazyka Python.
- `-r (--rules)` je povinný prepínač, ktorý uvádza cestu k súboru s reálnou sadou pravidiel. Uvedená reálna sada pravidiel sa v analyzátore spracuje a počítá sa z nej výstupný parametrový súbor formátu ClassBench. Formát parametrového súboru nástroja ClassBench je podrobne rozobraný v sekcii 3.2.1.
- `-f (--format)` je povinný prepínač, ktorý uvádza cestu k súboru s definíciou formátu pravidiel v reálnej sade uvedenej prepínačom '`-r`'. Všetky potrebné znalosti

pre návrh definície formátu pravidiel prijateľných implementovaným analyzátorom sa nachádzajú v podkapitole 4.3.

- `-o (--output)` je voliteľný prepínač, ktorým je možné uviesť cestu k novo vytvorenému výstupnému súboru. Do výstupného súbora analyzátor uloží vypočítané parametre. Aby uloženie výstupu do súboru prebehlo úspešne, súbor uvedený cestou nesmie existovať, alebo musí byť aspoň prázdny. V prípade, že je analyzátor spustený bez tohto prepínača, vypočítané parametre sa vypíšu do konzoly na štandardný výstup.
- `-l (--logs)` je voliteľný prepínač, ktorý zapne výpis varovných hlásení na štandardný chybový výstup pri spracovaní položiek pravidiel z reálnej sady do pomocných dátových štruktúr programu. Varovné hlásenie sa vypíše v prípade, ak nejaké pravidlo zo sady nerešpektuje formát pravidiel. Čo to znamená, že pravidlo nerešpektuje formát, je vysvetlené v podkapitole 5.2.
- `-h (--help)` je voliteľný prepínač. Pri spustení analyzátora s týmto prepínačom sa do konzoly vypíše nápoveda programu.

Príklad príkazu spustenia implementovaného analyzátora:

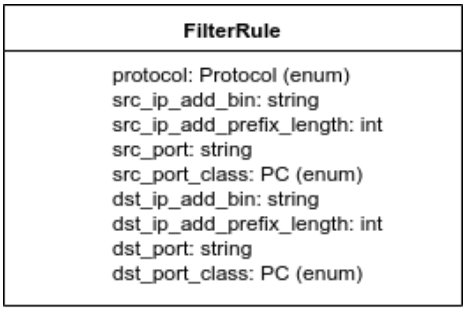
```
python3 -m filter_rule_analyzer -r rules.txt -f format.txt -o output.txt -l
```

Uvedený príkaz spúšťa analýzu reálnej sady pravidiel uloženej v súbore 'rules.txt'. Naplnenie pomocných dátových štruktúr z užitočných položiek pravidiel sady prebieha na základe formátu uloženého na prvom riadku v súbore 'format.txt'. Pri spracovaní položiek pravidiel sú zapnuté varovné hlásenia (prepínač '`-l`'). Vypočítané parametre sa uložia do súboru 'output.txt'.

5.2 Naplnenie dátových štruktúr analyzátora

Táto časť implementácie rieši problém univerzálnosti analyzátora rozoberaný v úvode kapitoly 4. Pravidlá sú v analyzátore reprezentované triedou `FilterRule`. Dátové členy zmienenej triedy predstavujú užitočné položky pravidiel a sú zobrazené na obrázku 5.1. IP adresy sú v dátových členoch triedy uložené v binárnom formáte, a to z dôvodu vytvárania binárnych prefixových stromov pri výpočte parametrov týkajúcich sa prefixov. Okrem hodnôt užitočných položiek pravidiel sú v dátových členoch triedy uložené aj triedy zdrojových a cieľových portov reprezentované výčtovým typom. Rozdelenie portov a rozsahov portov do tried sa nachádza v podkapitole 2.3. Bezparametrový konštruktor triedy `FilterRule` vytvára inštanciu, ktorej všetky dátové členy, predstavujúce užitočné položky pravidiel, majú wildcard hodnoty. Aby inštancia obsahovala aj iné ako wildcard hodnoty, je potrebné nad inštanciou volať setter metódy s parametrami hodnôt užitočných položiek. V ďalšom texte je volanie setter metód s parametrami hodnôt užitočných položiek nad inštanciou triedy `FilterRule` zjednodušené pod pojmom naplnenie `FilterRule` inštancie.

Vytvorenie a naplnenie inštancie triedy `FilterRule` pre každé pravidlo v reálnej sade má v implementovanom analyzátore na starosti funkcia statickej triedy `FilterRuleGenerator` pomenovaná `create_generator()`. Naplnenie inštancií je riadené formátom pravidiel v súbore uvedeným prepínačom '`-f`'. V prípade, ak pravidlo zo sady vyhovuje formátu, sú z neho vyextrahované užitočné položky do dátových členov inštancie triedy `FilterRule`. Položkami naplnená inštancia je z funkcie navrátená kľúčovým slovom `yield`. Týmto je



Obr. 5.1: Dátové členy triedy FilterRule

vo funkcii `create_generator()` vytvorený generátor `FilterRule` inštancií. Generátor je v jazyku Python objekt, ktorého položky sú iterovateľné `foreach` cyklom. Ďalšou výhodou generátora je, že v pamäti je vždy uložená iba aktuálna položka, na ktorú ukazuje iterátor, a nie všetky položky generátora [10].

Pseudokód funkcie `create_generator()` pre vytvorenie generátora `FilterRule` inštancií sa nachádza v algoritme 1. Opisovaná funkcia má 3 parametre: generátor riadkov zo súboru s reálnou sadou pravidiel, reťazec s formátom pravidiel a boolean premennú `is_stderr_on`, ktorá rozhoduje o tom, či má byť zapnutý varovný výpis hlásení pri spracovaní slov zo súboru s reálnou sadou pravidiel. Funkcia obsahuje 3 cykly, ktoré riadia vytvorenie generátora.

V prvom (nevnorenom) cykle sú prechádzané riadky z generátoru riadkov reálnej sady pravidiel. Predpokladá sa, že súbor s reálnou sadou obsahuje jednotlivé zápisy pravidiel na osobitných riadkoch. V tele tohto cyklu je najprv bezparametrovým konštruktorom vytvorená nová inštancia triedy `FilterRule`. Ďalej sú inicializované pomocné premenné.

V druhom (raz vnorenom) cykle sú prechádzané jednotlivé slová zo zápisu pravidla. Zápis pravidla je daný predchádzajúcim nevnoreným cyklom. Slovám zo zápisov pravdiel sa pri ich spracovaní v programe priradzujú typy. Možné typy slov zo zápisu pravidiel sú: any, protokol, IP adresa, port, číslo, kľúčové slovo alebo wildcard. Typ any je určený pre slová ako 'any', 'all alebo znak '*'. Typ wildcard je priradený slovu vo chvíli, ak nevyhovuju žiadnemu inému typu. Vo funkcii je typ spracovávaného slova uložený v premennej `real_type`. Napríklad pre slovo 'TCP' zo zápisu pravidla sa do premennej `real_type` uloží typ protokol. Pre slovo zo zápisu pravidla sa musí nájsť jeho definícia vo formáte. To sa deje v treťom cykle, kde sú prechádzané slová zo zápisu formátu.

V tele tretieho (dvakrát vnoreného) cyklu je do premennej `expected_type` uložené slovo zo zápisu formátu pravidiel na indexe daného premennou `format_index`. Slovo zo zápisu formátu pravidiel predstavuje parameter alebo kľúčové slovo z navrhnutého spôsobu zápisov formátov. Po uložení slova z formátu do premennej `expected_type` môžu ďalej nastať 3 možnosti:

- Slovo z formátu vyhovuje typu aktuálne spracovávaného slova zo zápisu pravidla. Typ slova zo zápisu pravidla je daný predchádzajúcim raz vnoreným cyklom (je uložený v premennej `real_type`). Napríklad vyhovujúcim typom slova zo zápisu pravidla pre parameter SRC_IP (zdrojová IP adresa) je typ any (napríklad slovo 'all') alebo typ IP adresa (napríklad slovo '10.0.0.0/24'). Pokiaľ slovo z formátu vyhovuje aktuálne spracovávanému slovu zo zápisu pravidla a spracovávané slovo je užitočnou položkou pravidla, potom je príslušný dátový člen `FilterRule` inštancie, vzniknutej v prvom cykle, naplnený hodnotou tejto položky. Index prechádzaných slov z formátu pravidiel

je po naplnení inštancie navýšený o hodnotu 1 a z cyklu prechadzajúceho cez slová formátu sa vyskočí, aby sa mohlo spracovať ďalšie slovo zo zápisu pravidla.

- V prípade, ak slovo z formátu nevyhovuje typu aktuálne spracovávaného slova z pravidla a ide o povinné slovo vo formáte pravidiel (na konci slova formátu sa nenachádza znak '?'), premenná `error` sa nastaví na hodnotu true a z tela cyklu sa vyskakuje. Táto možnosť nastane napríklad, ak slovo z formátu uložené v premennej `expected_type` je 'PROTOCOL' a typ spracovávaného slova, uloženého v premennej `real_type`, je port.

- Posledná možnosť nastane, ak slovo z formátu nevyhovuje typu aktuálne spracovávaného slova z pravidla a ide o voliteľné slovo vo formáte pravidiel (na konci slova formátu sa nachádza znak '?'). Táto možnosť nastane napríklad, ak slovo z formátu uložené v premennej `expected_type` je 'PROTOCOL?' a typ spracovávaného slova zo zápisu pravidla je port. V tomto prípade sa index prechádzaných slov z formátu pravidiel navýši o 1 a v cykle prechadzajúceho cez slová formátu sa pokračuje. Hľadá sa ďalší parameter alebo kľúčové slovo z formátu, ktoré by mohlo vyhovovať aktuálne spracovávanemu slovu zo zápisu pravidla.

Po prechode cez všetky slová v zápise formátu je položkami naplnená inštancia triedy `FilterRule` navrátená kľúčovým slovom `yield`. Ak sa pri spracovaní slov zápisu pravidla vyskytla chyba, je potom aj čiastočne naplnená inštancia ignorovaná. Jedinou chybou, ktorá vo funkcii môže nastať je, že sa v zápise pravidla nevyskytol očakávaný typ slova na mieste povinného parametru alebo povinného kľúčového slova vo formáte. Ak je zapnutý výpis varovných hlásení, na štandardný chybový výstup sa vypíše, aký povinný typ slova z formátu sa v zápise pravidla nevyskytol.

5.3 Výpočet a výpis parametrového súbora

Každý parameter z parametrového súboru formátu ClassBench má v implementovanom analyzátore svoju vlastnú triedu. Výnimkou sú akurát parametre počítané z binárných stromov (rozdelenie pravdepodobností vetvenia uzlov, priemer skew hodnôt na úrovňach stromu a parameter prahu zanorenia prefixov). Tieto parametre nemajú vlastnú triedu, ale iba jednu spoločnú triedu pomenovanú `TrieDistributions`. Trieda reprezentujúca parameter parametrového súbora musí implementovať rozhranie `IParameter`. Toto rozhranie má 3 abstraktné funkcie, ktoré musí deffinovať každá trieda implementujúca toto rozhranie. Obecné popisy funkcií rozhrania sa nachádzajú v nasledujúcom zozname:

- **`extract_data(self, filter_rule)`** ide o funkciu, ktorá z jednej inštancie triedy FilterRule (parameter `filter_rule`) extrahuje všetky dáta potrebné pre výpočet parametru a uloží ich v inštančných premenných triedy. Aký parameter sa počíta je dané triedou. Napríklad v triede `PortDistribution` sa počíta parameter rozdelenia pravdepodobnosti hodnôt portov alebo rozsahov portov.

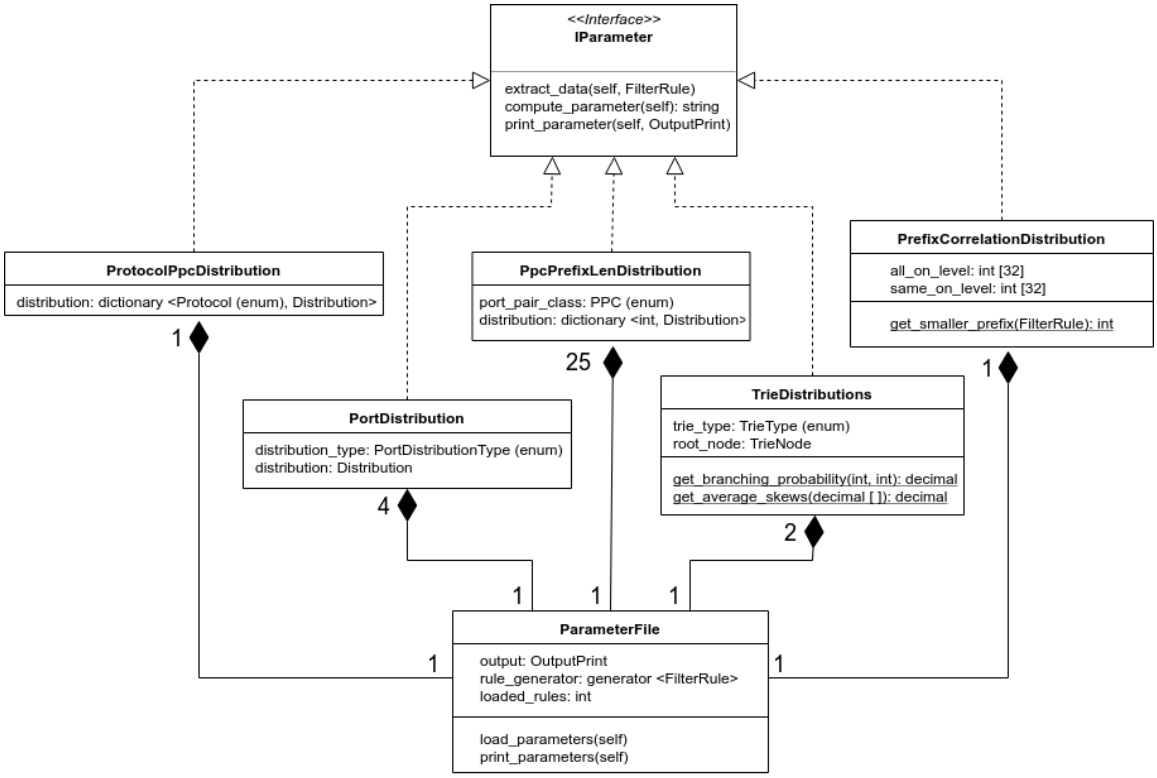
- **`compute_parameter(self)`** ide o funkciu, ktorá počíta parameter z dát uložených v inštančných premenných triedy.

- **`print_parameters(self, output)`** ide o funkciu, ktorá vypisuje vypočítaný parameter v ClassBench formáte na výstup programu. O tom, či výstup ide do súboru alebo na štandardný výstup rozhoduje parameter funkcie `output`.

Algorithm 1 Pseudokód funkcie pre vytvorenie generátora FilterRule inštancií

```
1: procedure CREATE_GENERATOR(rule_generator, rule_format, is_stderr_on)
2:     format_parts_count ← len(rule_format)
3:     for rule in rule_generator do
4:         filter_rule ← FilterRule()
5:         format_index ← 0
6:         error ← false
7:         for rule_word in rule do
8:             real_type ← get_type(rule_word)
9:             while format_index != format_parts_count and error == false do
10:                expected_type ← rule_format[format_index]
11:                if is_suitable(real_type, expected_type) then
12:                    filter_rule.fill_data_member(rule_word, expected_type)
13:                    format_index ← format_index + 1
14:                    break
15:                else if is_mandatory(expected_type) then
16:                    error ← true
17:                    break
18:                else if is_optional(expected_type) then
19:                    format_index ← format_index + 1
20:                end if
21:            end while
22:        end for
23:        if error == false then
24:            yield filter_rule
25:        else if is_stderr_on == true then
26:            print_warning(expected_type)
27:        end if
28:    end for
29: end procedure
```

Všetky triedy potrebné pre výpočet parametrového súboru sa nachádzajú na obrázku 5.2. Z diagramu je vidieť, že inštancie tried implementujúcich rozhranie `IParameter` (ďalej nazývané ako inštancie parametrových tried) sa nachádzajú v dátových členoch triedy `ParameterFile`. Trieda `ParameterFile` reprezentuje celý parametrový súbor. Okrem inštancií parametrových tried sa v dátových členoch zmienenej triedy nachádza generátor `FilterRule` inštancií, inštancia triedy `OutputPrint` a premenná určujúca počet úspešne spracovaných inštancií z generátora. Implementácii generátora `FilterRule` inštancií bola venovaná predchádzajúca podkapitola 5.2. Inštancia `OutputPrint` rozhoduje o výpise výstupu analyzátora do konzoly alebo do súboru. Trieda `ParameterFile` má iba 2 funkcie: `load_parameters(self)` a `print_parameters(self)`.



Obr. 5.2: UML diagram tried potrebných pre výpočet parametrového súbora

Funkcia `load_parameters(self)` napĺňa inštancie parametrových tried pre výpočet parametrov potrebnými dátmi volaním funkcie `extract_data(self, filter_rule)` s parametrom `FilterRule` inštancie danej z prechodu generátorom. Pseudokód funkcie sa nachádza v algoritme 2. Premenná `parameter_list` je zoznam všetkých inštancií parametrových tried v triede `ParameterFile`. Po spracovaní `FilterRule` inštancie sa premenná s počtom načítaných inštancií navýši o jednotku. Je potrebné poznamenať, že v tejto časti kódu sa už pri spracovaní nepredpokladá žiadna chyba, pretože sa pracuje iba s FilterRule inštanciami, ktoré boli validne vytvorené zo súboru s reálnou sadou a nachádzajú sa v generátore.

Funkcia `print_parameters(self)` volá v správnom poradí pre všetky inštancie parametrových tried funkciu `print_parameter(self, output)`, ktorá vypíše parametre na výstup daný inštančnou premennou triedy `output`. V tejto funkcii prebieha výpočet parametrov volaním funkcie `compute_parameter(self)`. Pseudokód funkcie `print_parameters(self)` sa nachádza v algoritme 3.

Algorithm 2 Pseudokód funkcie pre naplnenie inštancií parametrových tried, pre výpočet parametrov, potrebnými dátmi

```
1: procedure LOAD_PARAMETERS(self)
2:     for filter_rule in rule_generator do
3:         for parameter in parameter_list do
4:             parameter.extract_data(filter_rule)
5:             loaded_rules = loaded_rules + 1
6:         end for
7:     end for
8: end procedure
```

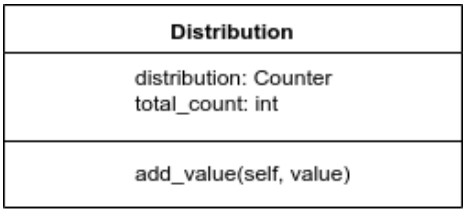
Algorithm 3 Pseudokód funkcie pre výpis vypočítaných parametrov

```
1: procedure PRINT_PARAMETERS(self)
2:     for parameter in parameter_list do
3:         parameter.print_parameter(self.output)
4:     end for
5: end procedure
```

5.3.1 Parametre rozdelenia pravdepodobnosti hodnôt

Parametrové triedy `ProtocolPpcDistribution`, `PortDistribution` a `PpcPrefixLenDistribution` počítajú parametre týkajúce sa rozdelenia pravdepodobnosti hodnôt špecifikovaných triedou. Pre počítanie rozdelenia pravdepodobnosti hodnôt majú triedy vo svojich dátových členoch vždy nejak zakompovanú inštanciu triedy `Distribution`. Členovia triedy `Distribution` sú zobrazení na obrázku 5.3. Ich popis je uvedený v nasledujúcom zozname:

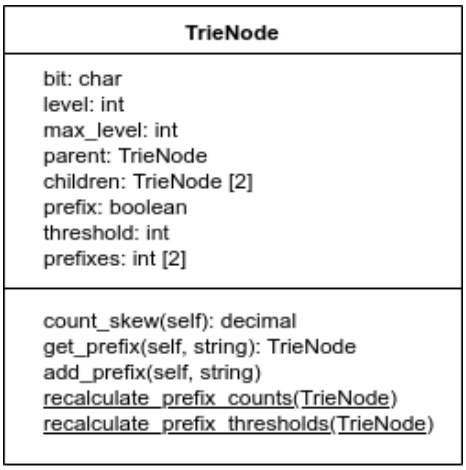
- `distribution` ide o kolekciu typu `Counter` importovanú zo štandardných kolekcií Pythonu [13]. Kolekcia v sebe udržiava dvojice: (unikátna hodnota kolekcie, počet výskytov unikátnej hodnoty v kolekcii). Napríklad dvojica ('TCP', 6) znamená, že hodnota 'TCP' sa v kolekcii nachádza 6-krát.
- `total_count` ide o premennú, určujúcu celkový počet hodnôt v kolekcii `distribution`. Ak sú počty výskytov jednotlivých unikátnych hodnôt vydelené celkovým počtom hodnôt (premenná `total_count`) vzniká rozdelenie pravdepodobnosti hodnôt.
- `add_value(self, value)` ide o funkciu, ktorá pridáva hodnotu do `distribution`. Pri pridávaní hodnoty do kolekcie môžu nastať 2 možnosti. Hodnota sa v kolekcii ešte nevyskytovala – potom sa vytvára nová dvojica, kde hodnota bude mať v dvojici jeden výskyt. Alebo sa hodnota v kolekcii už vyskytla – potom je počet výskytov tejto hodnoty v už existujúcej dvojici navýšený o jednotku. Celkový počet hodnôt v kolekcii `total_count` je navýšený o jednotku pri obidvoch prípadoch.



Obr. 5.3: Trieda použitá pre výpočet rozdelenia pravdepodobnosti hodnôt

5.3.2 Parametre binárneho prefixového stromu

Parametre počítané z binárneho prefixového stromu (rozdelenie pravdepodobností vetvenia uzlov, priemer skew hodnôt na úrovňach stromu a parameter prahu zanorenia prefixov) sú v implementovanom analyzátore reprezentované triedou `TrieDistributions`. Binárny prefixový strom je dátová štruktúra zložená zo zápisov prefixov v binárnej forme. Podrobnejšie sa binárnemu prefixovému stromu sa venuje sekcia 3.2.1. V spomínanej sekcii je graficky znázornený príklad stromu a uvedené, ako sa z neho matematicky počítajú vyžadované parametre. Elementárne častí stromu, ktoré tvoria prefixy stromu sa nazývajú uzly. Uzol stromu je v implementovanom analyzátore reprezentovaný triedou `TrieNode`. Členovia tejto triedy sú zobrazení na obrázku 5.4. Ich popis sa nachádza v nasledujúcich odsekoch.



Obr. 5.4: Trieda reprezentujúca uzol v binárnom prefixovom strome

Okrem jedného špeciálneho uzlu, nazývaného koreň stromu, majú všetky uzly stromu v dátovom člene `bit` uloženú hodnotu 0 alebo 1. Na ktorom indexe zápisu prefixu (na ktorej úrovni stromu) sa bit nachádza, určuje člen `level`. Najväčší index zápisov prefixov v strome (maximálna úroveň stromu) je uložený iba v člene koreňa stromu `max_level`. Tento údaj je použitý pri výpočte a výpise parametrov stromu.

Všetky uzly okrem koreňa stromu majú svojeho rodiča. Rodič uzla reprezentuje predchádzajúci bit v stromovom zápise prefixu a je uložený v člene `parent`. Uzly v strome môžu mať maximálne 2 potomkov. Potomkovia uzlu sú uzly reprezentujúce možné nasledujúce bity v stromovom zápise prefixu. Potomkovia uzlu sa nachádzajú v zozname `children`. Pri výpočte rozdelenia pravdepodobností vetvenia uzlov sa práve z týchto zoznamov na jednotlivých úrovňach stromu zisťujú potrebné počty uzlov s 1 potomkom a s 2 potomkami.

Aby bolo možné prechodom cez uzly stromu získať binárny zápis prefixu, je potrebné vedieť, z ktorého uzlu pri prechode vychádzať a v ktorom uzle zápis prefixu končí. Uzol stromu, z ktorého vychádzajú všetky zápisy prefixov v strome, je už spomínaný koreň stromu. Uzlov, v ktorých sa zápisy prefixov pri prechode stromom končia, môže byť viac. V ďalšom texte sú tieto uzly nazývané ako prefixové uzly. V implementovanom strome sú prefixové uzly od ostatných odlišené tým, že majú nastavený dátový člen `prefix` na hodnotu true. Dátový člen uzla `threshold` udáva maximálny počet prefixových uzlov od potomkov uzla až ku listom stromu. Hodnota tohto člena v koreni stromu potom predstavuje parameter prahu zanorenia prefixov (prefix nesting threshold).

Počet zápisov prefixov stromu končiach v ľavej a pravej časti podstromu uzlu je uložený v dvojprvkovom poli `prefixes`. V prípade, že sa v podstrome uzlu nachádza jediný prefixový uzol, to automaticky neznamená, že počet zápisov prefixov končiacich v podstrome tohto uzlu je 1. Ak sa v sade zápisov prefixov tvoriacich strom nachádza viacero rovnakých zápisov, je možné, že v podstrome vyšetrovaného uzla je počet zápisov prefixov končiacich v prefixovom uzle väčší ako 1. Hodnoty v poli `prefixes` sú použité vo funkcii triedy `count_skew(self)` pre výpočet skew hodnoty uzla.

Funkcie `get_prefix(self, prefix)` a `add_prefix(self, prefix)` sú volané iba koreňovým uzlom stromu. Prvá funkcia umožňuje získať prefixový uzol stromu zo zápisu prefixu daného parametrom. Druhá funkcia do stromu pridáva nové uzly alebo aktualizuje hodnoty dátových členov, v strome už nachádzajúcich sa uzlov. Dátové členy sú aktualizované viacnásobným volaním statických funkcií: `recalculate_prefix_counts(node)` a `recalculate_prefix_thresholds(node)` s parametrom každého uzlu tvoriacieho zápis pridávaného prefixu.

5.3.3 Parameter korelácie prefixov

Poslednou parametrovou triedou potrebnou pre výpočet parametrového súboru je `PrefixCorrelationDistribution`. Trieda má za úlohu počítať parameter prefixovej korelácie. Parameter určuje pravdepodobnosti, že binárne zápisy zdrojových a cieľových prefixov pravidiel uložených v dátových členoch inštancií triedy `FilterRule` z generátora sa nemenia pre jednotlivé úrovne dĺžok prefixov od 1 až po úroveň stanovenú menším z prefixov.

Pseudokód funkcie `extract_data(self, filter_rule)`, ktorá napĺňa inštančné premenné tejto parametrovej triedy, sa nachádza v algoritme 4. Inštancia triedy `FilterRule`, ktorej zdrojový alebo cieľový prefix má wildcard hodnotu, je ignorovaná. Pred cyklom, ktorý prechádza jednotlivými bitmi zo zápisu zdrojového prefixu, sa najprv zistí veľkosť menšieho z prefixov. Táto veľkosť je použitá pre orezanie zápisu zdrojového prefixu v prípade, ak by bol zápis cieľového prefixu menší ako zápis zdrojového prefixu. Vo `for` cykle sa prechádza zápis zdrojového prefixu po bitoch postupne od prvého bitu. Hodnota poľa `all_on_level` na indexe bitu `bit_index` je pri prechode cyklom vždy inkrementovaná o jednotku. Hodnota poľa `same_on_level` je inkrementovaná iba ak sú bity na úrovni prefixov zhodné. V prípade, že sú bity na úrovni prefixov rozdielne, porovnávanie končí a z tela cyklu sa vyskakuje. Popis naplňovaných dátových členov triedy sa nachádza v nasledujúcom zozname:

- `same_on_level:` Ide o pole celočíselných hodnôt o veľkosti 32. Hodnoty na jednotlivých indexoch poľa vyjadrujú počet `FilterRule` inštancií, ktorých zápisy prefixov sú rovnaké na úrovni prefixu daného indexom.
- `all_on_level:` Ide o pole celočíselných hodnôt o veľkosti 32. Hodnoty na jednotlivých indexoch poľa vyjadrujú počet `FilterRule` inštancií, ktorých zápisy prefixov boli

rovnaké na predchádzajúcej úrovni zápisu. Parameter prefixovej korelácie vzniká, ak sú počty z poľa `same_on_level` vydelené s počtami na rovnakých indexoch z poľa `all_on_level`.

Algorithm 4 Pseudokód funkcie, ktorá napĺňa dátové členy parametrovej triedy pre výpočet korelácie prefixov

```
1: procedure EXTRACT_DATA(self, filter_rule)
2:     if filter_rule.src_ip_add_bin == '*' or filter_rule.dst_ip_add_bin == '*'
   then
3:         return
4:     end if
5:     smaller_plength ← get_smaller_prefix(filter_rule)
6:     bit_index ← 0
7:     for bit in filter_rule.src_ip_add_bin[: smaller_plength] do
8:         all_on_level[bit_index] ← all_on_level[bit_index] + 1
9:         if bit == filter_rule.dst_ip_add_bin[bit_index] then
10:            same_on_level[bit_index] ← same_on_level[bit_index] + 1
11:        else
12:            break
13:        end if
14:        bit_index ← bit_index + 1
15:    end for
16: end procedure
```

5.4 Porovnanie procedurálnej a objektovo orientovanej implementácie

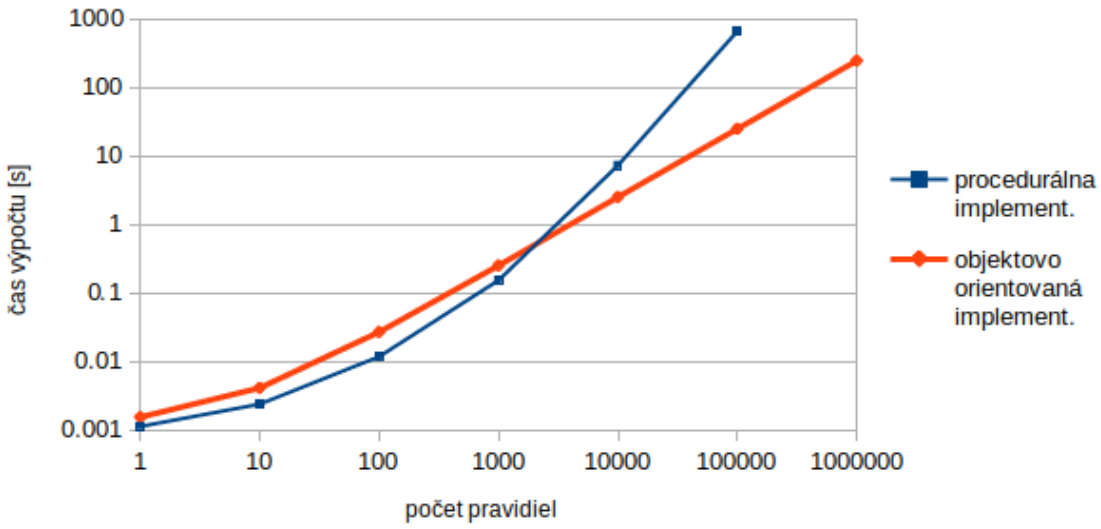
Prvá verzia analyzátora bola implementovaná procedurálnym spôsobom. Procedurálny spôsob znamená, že celý program sa skladá iba z funkcií, ktoré sa medzi sebou prevolávajú. Procedurálnym spôsobom sa napríklad programuje v jazyku C. Druhá, finálna verzia analyzátora bola implementovaná objektovo orientovaným spôsobom. Objektovo orientovaný spôsob spôsob znamená, že program sa skladá z objektov, ktoré reprezentujú modely konceptov, procesov alebo vecí z reálneho sveta [1]. Takýmito objektami sú v implementovanom analyzátore už v predchádzajúcich podkapitolách zmienené filtrovacie pravidlo (trieda `FilterRule`), parametrový súbor (trieda `ParameterFile`) alebo binárny prefixový strom (trieda `TrieNode`).

Porovnanie verzií analyzátorov je zamerané na časovú a pamäťovú náročnosť pri výpočte parametrového súboru zo sád náhodne vygenerovaných pravidiel s počtom pravidiel 10^n (n je prirodzené číslo a jeho maximálna hodnota pri meraní naročností bola 6). Pravidlá sady boli pre meranie náhodne vygenerované v pomocnom programe a mali jednoduchý formát. Zápis pravidiel sady sa skladal iba z položiek užitočných pre výpočet parametrového súbora formatu ClassBench. Meranie časovej a pamäťovej náročnosti výpočtov bolo prevádzané na stroji s nasledujúcou konfiguráciou:

- operačný systém Ubuntu 18.04

- procesor Intel Core i5-6200U
- pamäť RAM 8GB DDR4

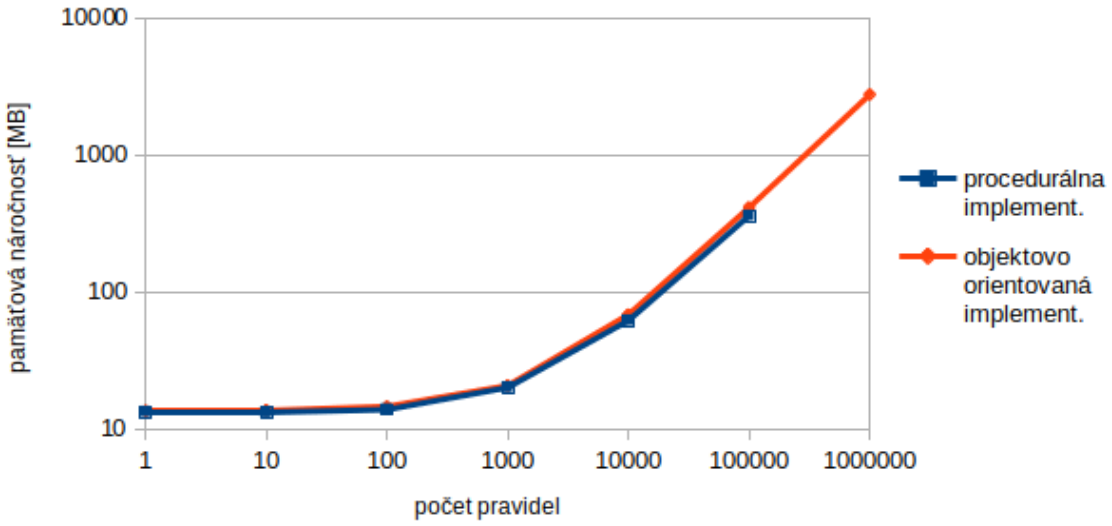
Časová náročnosť bola meraná jednoduchým spôsobom. Meranie bolo uskutočnené ako rozdiel 2 premenných, do ktorých bol zachytený aktuálny času (timestamp) pred výpočtom parametrov a po ich výpočte. Od premennej s časom po výpočte parametrov sa odpočítala premenná s časom pred výpočtom parametrov. Výsledkom bol čas potrebný pre výpočet parametrov. Porovnanie časovej náročnosti verzií analyzátorov je uvedené v grafe 5.5. Od počtu desaťtisíc spracovávaných pravidiel bolo pri meraní zistené, že výpočet parametrov objektovo orientovanej implementácie je rýchlejší ako u procedurálnej implementácie. Pre počty pod desaťtisíc pravidiel bola rýchlosť výpočtu parametrov trochu rýchlejšia u procedurálnej implementácie. Dôvodom, prečo je objektovo orientovaná implementácia pri vyšších počtoch pravidiel omnoho rýchlejšia je to, že nemusí viacnásobne prechádzať veľkú sadu pravidiel, aby získala dáta potrebné pre výpočet parametrov. Všetky dáta potrebné pre výpočet parametrov má táto implementácia uložené vo svojich objektoch a získala ich iba jedným prechodom cez sadu pravidiel. Procedurálna implementácia na rozdiel od objektovo orientovanej implementácie nemá objekty (ani globálne premenné), a preto aby získala dáta potrebné pre výpočet jednotlivých parametrov, musí sadu pravdiel prechádzať opakovane. Pre sadu s milión pravidlami nebolo pre procedurálnu implementáciu meranie náročností uskutočnené.



Obr. 5.5: Graf porovnania časovej naročnosti verzie analyzátora s procedurálnou implementáciou a verzie analyzátora s objektovo orientovanou implementáciou

Meranie pamäťovej náročnosti bolo realizované funkciou `memory_info().rss` z voľne dostupnej knižnice psutil [4]. Zmienená funkcia bola pri meraní pamäťovej náročnosti volaná vždy na konci programu po výpočte parametrov. Týmto spôsobom potom funkcia vrátila celkové množstvo pamäte, ktorú použil proces programu pre kompletný výpočet parametrov. Do pamäťovej náročnosti je teda započítaná aj réžia, nie príliš pamäťovo úsporného jazyka, Python. Pre toto porovnávanie pamäťovej náročnosti verzií analyzátorov réžia

Pythonu nevadí, pretože podmienky, s ktorými boli obidve verzie analyzátora spúšťané, boli rovnaké. Porovnanie pamäťovej náročností verzií analyzátorov je uvedené v grafe 5.5. Pre všetky merania mala o trochu menšiu pamäťovú náročnosť procedurálna implementácia. Toto zistenie sa dalo očakávať, keďže procedurálna implementácia obsahuje iba funkcie, ktoré zo sád pravidiel extrahujú a vracajú pre výpočet parametrov potrebné dáta a neobsahuje žiadne objekty ani globálne premenné. Takéto šetrenie pamäti sa však určite neoplatí. Pri meraní náročnosti so stotísic pravidlami procedurálna implementácia ušetrila 54 MB pamäti, no jej výpočet trval o 10 minút dlhšie ako u objektovo orientovanej implementácie.



Obr. 5.6: Graf porovnania pamäťovej naročnosti verzie analyzátora s procedurálnou implementáciou a verzie analyzátora s objektovo orientovanou implementáciou

Kapitola 6

Analýza reálnych sád klasifikačných pravidiel

V kapitole 6 sa popisuje v práci vykonaná analýza klasifikačných pravidiel. Pre analýzu bolo k dipozícií celkom 8 sád klasifikačných pravidiel. Jedna z dostupných sád však nemohla byť analyzovaná, pretože položky jej pravidiel obsahovali IPv6 adresy, ktorých analýzu nástroj nepodporuje. O sadách bol dostupný iba ich názov a dátum získania. Neboli poskytnuté žiadne informácie o tom, z akého konkrétneho zariadenia na sieti boli sady získané. Prehľad analyzovaných sád s počtom ich pravidiel a dátumom ich získania sa nachádza v tabuľke 6.1.

Názov sady	Počet pravidiel	Dátum získania
fw	58	1.február 2008
fw2	201	1.február 2008
fw3-dump	125	1.február 2008
fw-dump	129	1.február 2008
part-hp	78	1.február 2008
fw-fast	105	2.február 2008
fw-hp	174	11.január 2009

Tabuľka 6.1: Analyzované sady klasifikačných pravidiel

Pre každý formát sady z uvedenej tabuľky bolo možné nadefinovať formát prijateľný implementovaným nástrojom a vygenerovať parametrový súbor formátu ClassBench [17]. Analýza bola v poradí vykonaná na nasledujúcich hodnotách položiek klasifikačných pravidiel: protokol, zdrojové a cieľové porty, zdrojové a cieľové prefixy. Vygenerované parametrové súbory boli užitočné hlavne pri analýze hodnôt prefixov, kde z nich nebol problém vyextrahovať rozdelenie pravdepodobnosti vetvenia uzlov na jednotlivých úrovňach stromov a priemer skew hodnôt na jednotlivých úrovňach stromov. Pri každej čiastočne prevedenej analýze na nejakej položke pravidiel, je v krátkosti uvedené porovnanie s obdobnou analýzou vykonanou v článku o nástroji ClassBench-ng [9].

6.1 Analýza protokolov a portov

Rozdelenie pravdepodobnosti hodnôt protokolov v analyzovaných sadách je uvedené v tabuľke 6.2. Hodnota wildcard protokolu mala prevažné zastúpenie v 4 zo 7 sád. V 3 ostatných sadách dominovalo zastúpenie TCP protokolu. Oproti analýze protokolov prevedenej v článku o ClassBench-ng je možné v tejto analýze sledovať nárast výskytov wildcard hodnôt protokolov a zníženie výskytov TCP protokolov.

Názov sady	wildcard	ICMP	TCP	UDP
fw	0.59	0.00	0.41	0.00
fw2	0.21	0.05	0.54	0.20
fw3-dump	0.88	0.00	0.07	0.05
fw-dump	0.79	0.01	0.16	0.04
part-hp	0.32	0.04	0.49	0.15
fw-fast	0.30	0.02	0.46	0.22
fw-hp	0.93	0.00	0.05	0.02

Tabuľka 6.2: Rozdelenie pravdepodobnosti hodnôt protokolov v analyzovaných sadách

Rozdelenie pravdepodobností tried zdrojových portov a tried cieľových portov v analyzovaných sadách je uvedené v tabuľkách 6.3 a 6.4. Analýza zdrojových portov nepriniesla žiadne prekvapenie. Zdrojové porty v analyzovaných sadách patrili až na pár výnimiek do triedy WC (wildcard). S rovnakým výsledkom ohľadom tried zdrojových portov prišla aj porovnávaná analýza v článku o ClassBench-ng. Cieľové porty v analyzovaných sadách tiež väčšinou patrili do triedy WC, ale nejedná sa tu o takú dominanciu ako u zdrojových portov, keďže trieda EM (exact match) tu má v priemere pätinové zastúpenie. Oproti porovnávanej analýze cieľových portov v článku o ClassBench-ng je možné sledovať nárast výskytov tried WC a pokles výskytov tried EM.

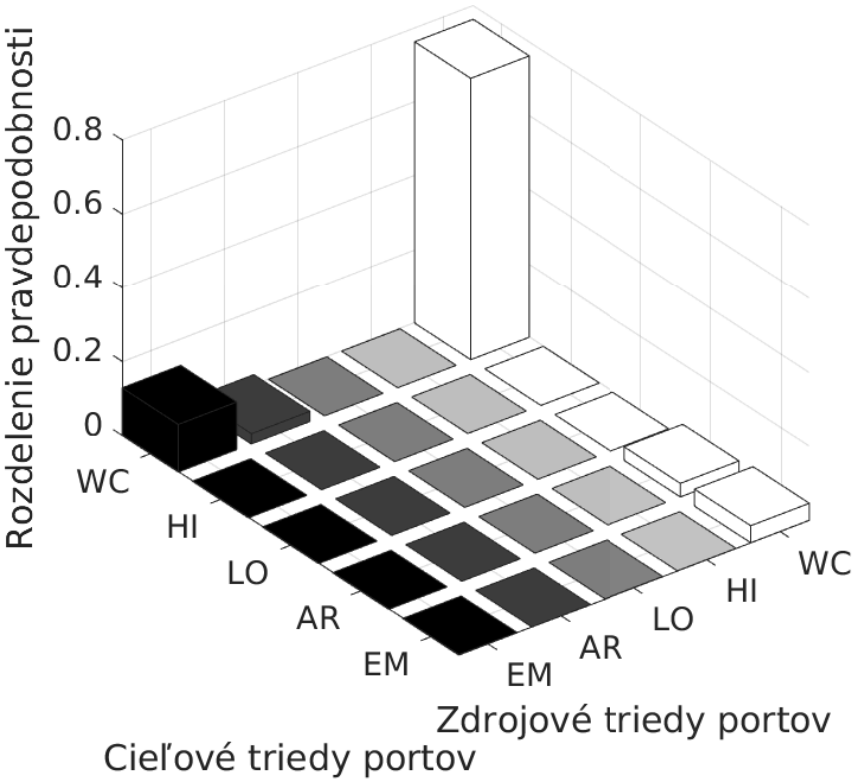
Názov sady	WC	HI	LO	AR	EM
fw	1.00	0.00	0.00	0.00	0.00
fw2	0.94	0.01	0.00	0.02	0.03
fw3-dump	1.00	0.00	0.00	0.00	0.00
fw-dump	1.00	0.00	0.00	0.00	0.00
part-hp	0.86	0.03	0.03	0.00	0.08
fw-fast	0.95	0.00	0.00	0.00	0.05
fw-hp	1.00	0.00	0.00	0.00	0.00

Tabuľka 6.3: Rozdelenie pravdepodobnosti tried zdrojových portov v analyzovaných sadách

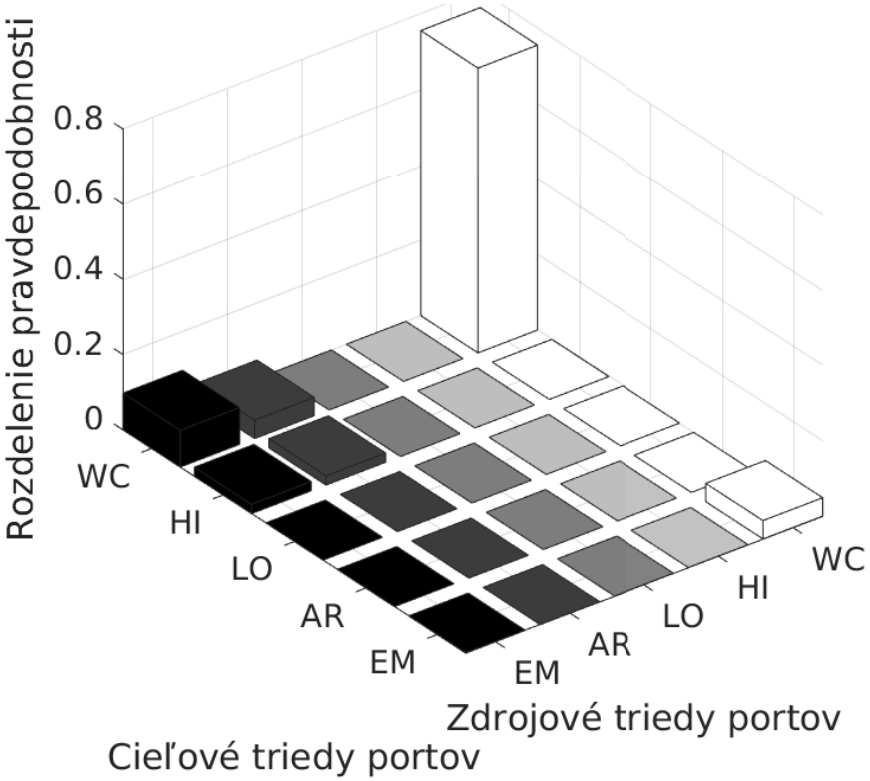
Názov sady	WC	HI	LO	AR	EM
fw	0.67	0.00	0.00	0.00	0.33
fw2	0.87	0.00	0.00	0.03	0.1
fw3-dump	0.9	0.00	0.00	0.02	0.08
fw-dump	0.81	0.00	0.00	0.01	0.18
part-hp	0.44	0.00	0.00	0.18	0.38
fw-fast	0.49	0.00	0.00	0.02	0.49
fw-hp	0.93	0.00	0.00	0.02	0.05

Tabuľka 6.4: Rozdelenie pravdepodobnosti tried cieľových portov v analyzovaných sadách

Matice PPC (port pair class) pre TCP protokol a UDP protokol z analyzovanej sady fw2 sú zobrazené na obrázkoch 6.1 a 6.2. Takýto dominantný výskyt PPC tried WC-WC bol v prevádzanej analýze s dostupnými sadami častým javom. Jedinými sadami, ktorých PPC matice pre jednotlivé protokoly sa od uvedených trochu odlíšili, boli sady fw, part-hp a fw-fast. V zmienených sadách sa častejšie vyskytovala PPC trieda WC-EM. Oproti porovnávanej analýze PPC tried v článku o ClassBench-ng je možné sledovať dominantný nárast výskytov PPC tried WC-WC. V porovnávanej analýze sa dokonca PPC triedy WC-WC nevyskytovali.



Obr. 6.1: PPC matica pre TCP protokol v sade fw2



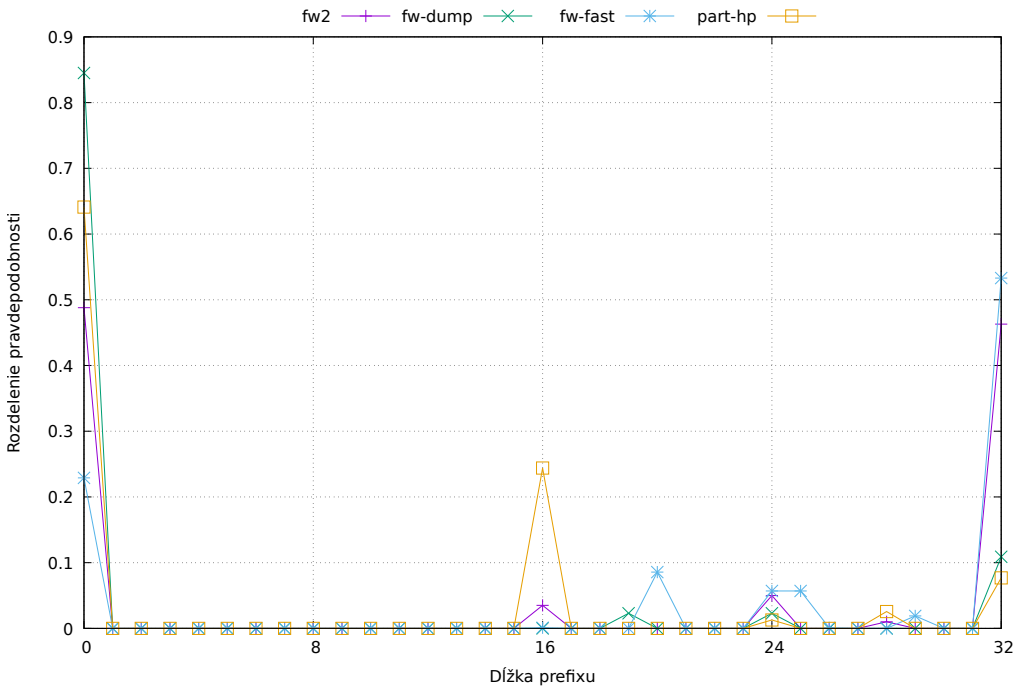
Obr. 6.2: PPC matica pre UDP protokol v sade fw2

6.2 Analýza prefixov

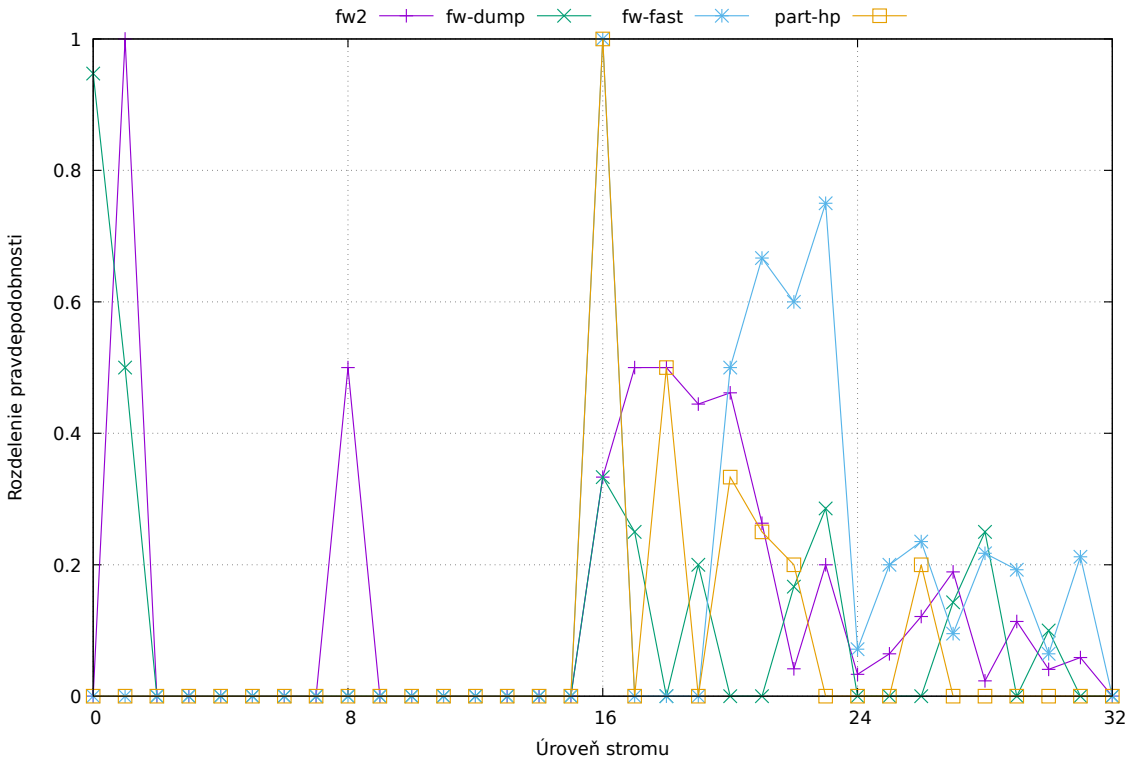
Grafy vytvorené pre analýzu prefixov sú: rozdelenie pravdepobodnosti dĺžok prefixov, rozdelenie pravdepodobnosti vetvenia uzlov stromu a priemery skew hodnôt stromu. Všetky zmienené grafy sú vytvárané osobitne pre zdrojové a cieľové prefixy vyskytujúce sa v klasifikačných pravidlách dostupných sád. Analýza zdrojových prefixov je zobrazená v grafoch na obrázkoch 6.3, 6.4, 6.5 a analýza cieľových prefixov je zobrazená v grafoch na obrázkoch 6.6, 6.7, 6.8. V grafoch nie sú zobrazené hodnoty pre všetky dostupné analyzované sady, pretože by sa graf stal neprehľadným. Namiesto toho sú do grafov vybraté 4 najviac jedinečné sady.

Najčastejšou nachádzajúcou sa dĺžkou prefixu v analyzovaných sadách pravidiel bola hodnota 0 (wildcard), 16, 24 a 32 (plne špecifikovaná adresa). V porovnávanej analýze v článku o ClassBench-ng to bola jednoznačne hodnota 24, ktorá dominovala dĺžkam prefixov.

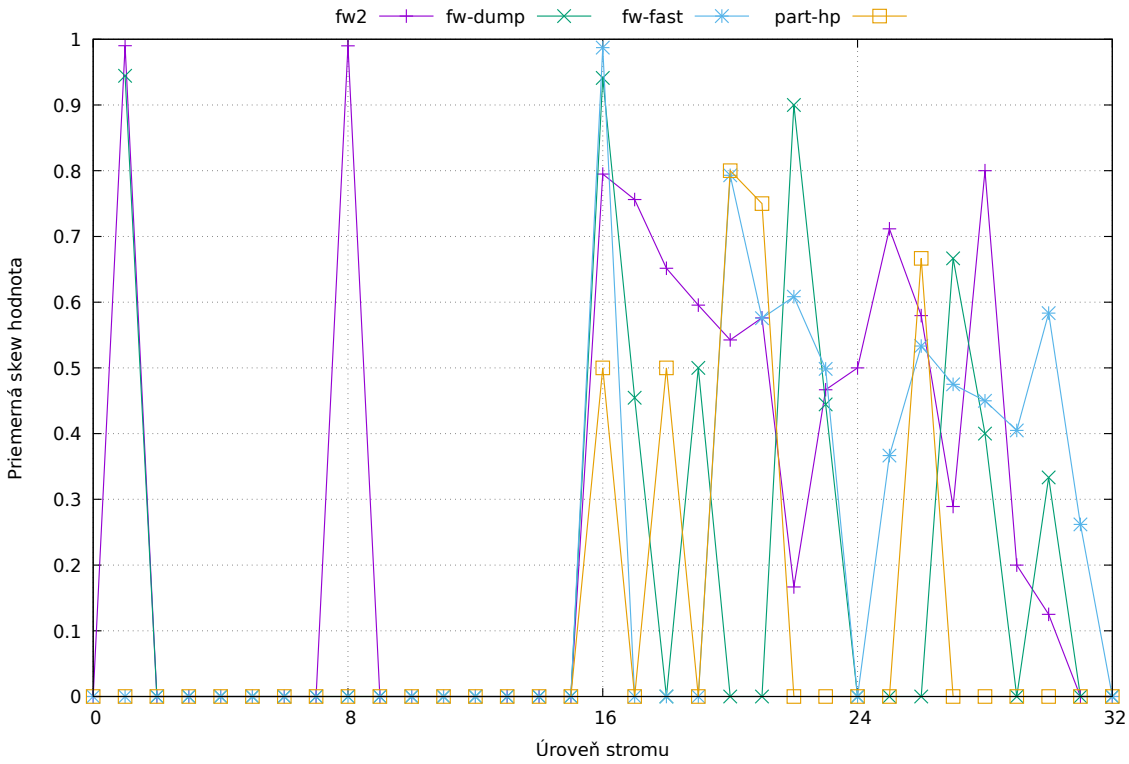
Zdrojový a aj cieľový prefixový strom zostrojený z prefixov v analyzovaných sadách pravidiel ukázal, že hlavne uzly od úrovne stromov 16 majú 2 potomkov. Priemer skew hodnôt na týchto úrovňach je často hodnota blížiaca sa k 1. Analýza prefixových stromov v článku o ClassBench-ng malá iné výsledky. Vetvenie uzlov od úrovne 16 postupne klesalo z hodnoty 1 až do úrovne 32 po hodnotu 0. Priemer skew hodnôt bola pre všetky úrovne hodnota okolo 0.5.



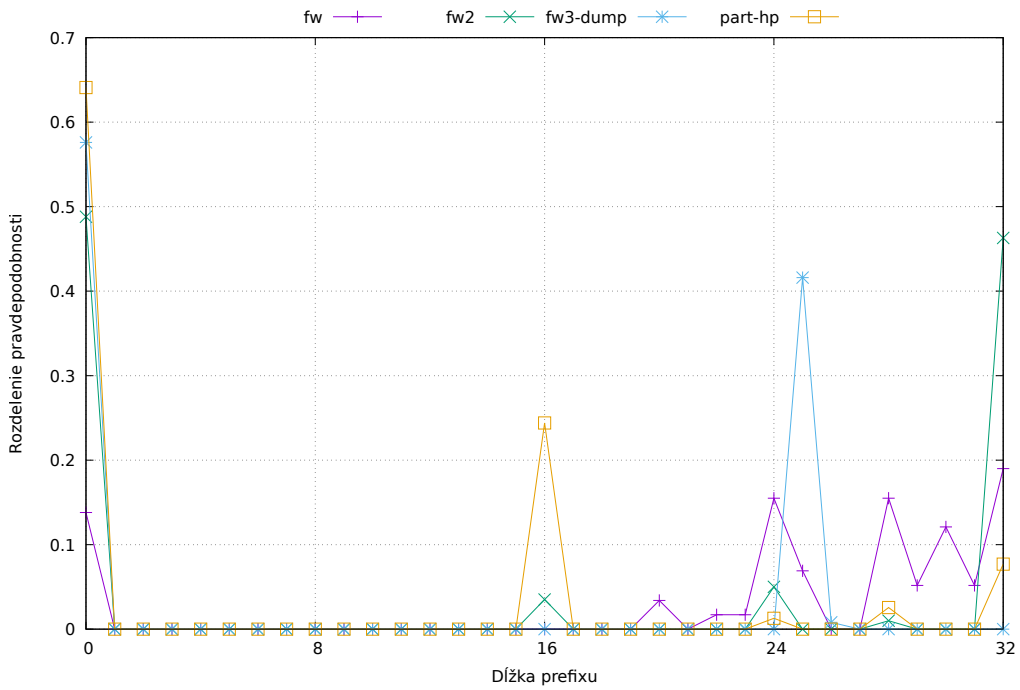
Obr. 6.3: Rozdelenie pravdepodobnosti dĺžok zdrojových prefixov analyzovaných sád



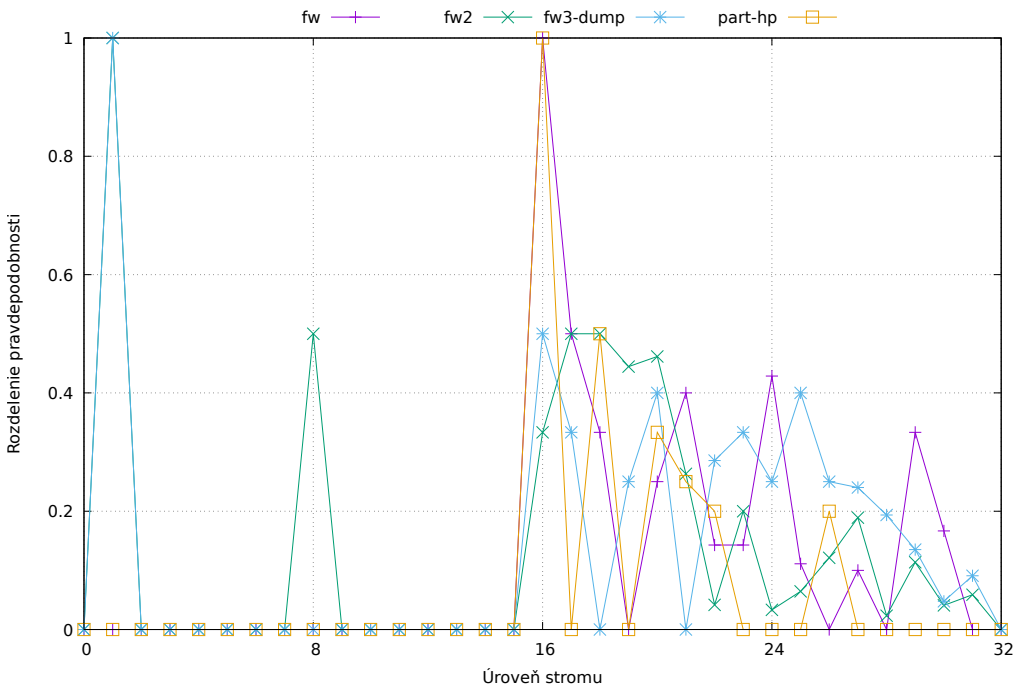
Obr. 6.4: Rozdelenie pravdepodobnosti vetvenia uzlov na jednotlivých úrovňach stromu zostrojeného zo zdrojových prefixov. Hodnota v bode na ose y vyjadruje pravdepodobnosť, že uzol na úrovni stromu má 2 potomkov. Doplnok tejto hodnoty do 1 predstavuje pravdepodobnosť, že uzol na úrovni stromu má iba 1 potomka.



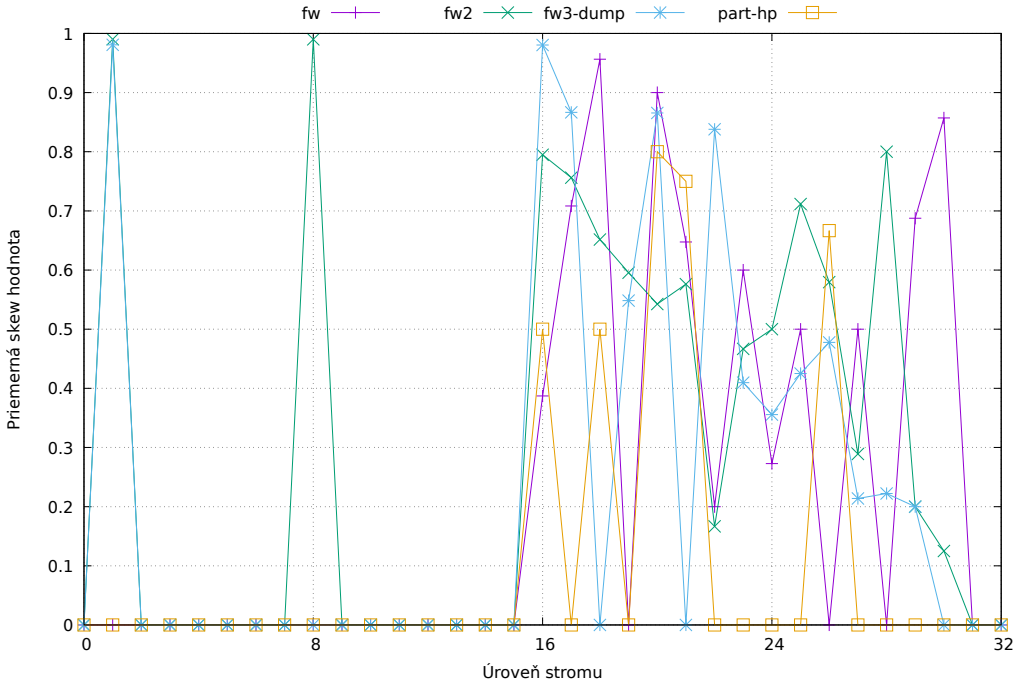
Obr. 6.5: Priemerná skew hodnota na jednotlivých úrovňach stromu zostrojeného zo zdrojových prefixov



Obr. 6.6: Rozdelenie pravdepodobnosti dĺžok cieľových prefixov analyzovaných sád



Obr. 6.7: Rozdelenie pravdepodobnosti vetvenia uzlov na jednotlivých úrovňach stromu zostrojeného z cieľových prefixov. Hodnota v bode na ose y vyjadruje pravdepodobnosť, že uzol na úrovni stromu má 2 potomkov. Doplnok tejto hodnoty do 1 predstavuje pravdepodobnosť, že uzol na úrovni stromu má iba 1 potomka.



Obr. 6.8: Priemerná skew hodnota na jednotlivých úrovňach stromu zostrojeného z cieľových prefixov

Kapitola 7

Záver

Cieľom práce bolo vytvoriť nástroj pre analýzu sád klasifikačných pravidiel. Výsledkom analýzy prevádzanej v nástroji mal byť presne špecifikovaný parametrový súbor umožňujúci generovanie umelých sád pravidiel v nástrojoch ClassBench [17] a ClassBench-ng [9]. Druhou požiadavkou pre cieľový nástroj bola jeho nezávislosť na formáte pravidiel analyzovaných sád. To znamená, že nástroj by nemal mať problém vytvoriť parametrový súbor zo sád klasifikačných pravidiel rôznych formátov používaných v odlišných nástrojoch v praxi. Implementovaný nástroj spĺňuje obidve požiadavky. Nástroju nerobí problém analyzovať reálne sady klasifikačných pravidiel ľubovoľných formátov a dokáže vytvárať parametrový súbor formátu nástroja ClassBench. Tento formát je možné bez zmien použiť aj v nástroji ClassBench-ng.

Pre splnenie prvej požiadavky, vytvorenie parametrového súboru, bolo v práci potrebné rozobrať zloženie parametrového súboru spomínaného nástroja a zistiť, ako sa jednotlivé parametre zo sád analyzovaných pravidiel počítajú. Implementácie algoritmov pre výpočet požadovaných parametrov vhodne využívajú pre svoju činnosť objekty jazyka Python, ktoré znižujú časovú a pamäťovú náročnosť výpočtov [10, 13].

Pre splnenie druhej požiadavky, nezávislosť na formáte analyzovaných pravidiel, boli v práci skúmané formáty klasifikačných pravidiel používaných v nástrojoch iptables [5], IPFW [3] a ovs-ofctl [12]. Na základe informácií získaných zo skúmania bol v práci navrhnutý spôsob definovania formátov analyzovaných sád. Formát nadefinovaný navrhnutým spôsobom popisuje, ako vyzerá jedno pravidlo v analyzovanej sade. Naplnenie pomocných dátových štruktúr programu, z ktorých sa v programe počítajú parametre, je riadené práve pomocou tohto formátu. Implementovaný nástroj je konzolová aplikácia a jej spustenie potom okrem prepínača uvádzajúceho cestu k súboru s analyzovanou sadou pravidiel vyžaduje uviesť aj druhý povinný prepínač uvádzajúci cestu k súboru s nadefinovaným formátom.

Nástroj sa ukázal ako užitočný pri záverečnej časti práce, analýze dostupných reálnych sád klasifikačných pravidiel. Pre 7 z 8 sád bolo možné nadefinovať ich formát a spustením nástroja získať zo sady parametrový súbor formátu nástroja ClassBench. Dôvodom, prečo z jednej sady nemohol byť vytvorený parametrový súbor, je to, že obsahovala IPv6 zápisy adries, ktoré nástroj nepodporuje. Rozšírenie nástroja by teda mohlo pridať podporu analýzy IPv6 adries. Ďalším užitočným rozšírením nástroja by bolo pridanie podpory pre analýzu položiek pravidiel protokolu OpenFlow [12]. Toto rozšírenie by umožnilo vytvorenie kompletného parametrového súboru formátu nástroja ClassBench-ng.

Literatúra

[1] Adobe Inc.: OBJECT-ORIENTED PROGRAMMING CONCEPTS: OBJECTS AND CLASSES. [Online; navštíveno 2.5.2019].
URL https://www.adobe.com/devnet/actionscript/learning/oop-concepts/objects-and-classes.html

[2] B. Claise, E.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, October 2004.
URL https://www.rfc-editor.org/rfc/rfc3954.txt

[3] FreeBSD: Manual for ipfw. 2018, [Online; navštíveno 1.5.2019].
URL https://www.freebsd.org/cgi/man.cgi?ipfw(8)

[4] Giampaolo Rodola: psutil documentation. [Online; navštíveno 3.5.2019].
URL https://psutil.readthedocs.io/en/latest

[5] Herve Eychenne: Man page for iptables. [Online; navštíveno 1.5.2019].
URL https://www.unix.com/man-page/linux/8/iptables

[6] IANA: Assigned Internet Protocol Numbers. 2017, [Online; navštíveno 12.12.2018].
URL
https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

[7] IANA: Service Name and Transport Protocol Port Number Registry. 2018, [Online; navštíveno 11.12.2018].
URL https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[8] IP Location: What is a TCP/IP? 2018, [Online; navštíveno 6.12.2018].
URL https://www.iplocation.net/tcp-ip

[9] J. Matoušek G. Antichi, A. Lučanský, A. W. Moore, J. Kořenek: ClassBench-ng: Recasting ClassBench After a Decade of Network Evolution. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications*, 2017, s. 204–216.

[10] Luciano Strika: Python's Generator Expressions: Fitting Large Datasets into Memory. 2018, [Online; navštíveno 2.5.2019].
URL https://towardsdatascience.com/pythons-list-generators-what-when-how-and-why-2a560abd3879

[11] Network Computing Editors: Six Benefits Of IPv6. 2011, [Online; navštíveno 10.12.2018].
URL
https://www.networkcomputing.com/networking/six-benefits-ipv6/1148014746

[12] Open vSwitch: Manual for ovs-ofctl. [Online; navštíveno 1.5.2019].
URL http://www.openvswitch.org/support/dist-docs/ovs-ofctl.8.txt

[13] Parul Pandey: Pythons Collections Module. 2019, [Online; navštíveno 1.5.2019].
URL https://towardsdatascience.com/pythons-collections-module-high-performance-container-data-types-cb4187afb5fc

[14] Rene Molenaar: Shortening IPv6 Addresses. 2016, [Online; navštíveno 9.12.2018].
URL https://networklessons.com/ipv6/shortening-ipv6-addresses

[15] studytonight: The TCP/IP Reference Model. [Online; navštíveno 9.12.2018].
URL
https://www.studytonight.com/computer-networks/tcp-ip-reference-model

[16] Střední průmyslová škola Brno: Sada protokolů TCP/IP. 2012, [Online; navštíveno 11.12.2018].
URL https:
//moodle.sspbrno.cz/pluginfile.php/6413/mod_resource/content/1/tcpip.pdf

[17] Taylor, D. E.; Turner, J. S.: ClassBench: A Packet Classification Benchmark. In *IEEE/ACM Transactions on Networking*, ročník 15, 2007, s. 499–511.

[18] Techopedia: OpenFlow. [Online; navštíveno 11.12.2018].
URL https://www.techopedia.com/definition/28935/openflow

[19] Výzkumná skupina ANT at FIT: Hardwarová akcelerace klasifikace paketu. Technická správa, 2010.

[20] Wikipedia: Adresa IP — Wikipedia, The Free Encyclopedia. 2018, [Online; navštíveno 6.12.2018].
URL http://sk.wikipedia.org/w/index.php?title=Adresa%20IP&oldid=6805398

[21] Wikipedia: Internet — Wikipedia, The Free Encyclopedia. 2018, [Online; navštíveno 6.12.2018].
URL http://sk.wikipedia.org/w/index.php?title=Internet&oldid=6823860

[22] Wikipedia: IPv6 — Wikipedia, The Free Encyclopedia. 2018, [Online; navštíveno 6.12.2018].
URL http://sk.wikipedia.org/w/index.php?title=IPv6&oldid=6791918

[23] Wikipedia: MAC adresa — Wikipedia, The Free Encyclopedia. 2018, [Online; navštíveno 10.12.2018].
URL http://sk.wikipedia.org/w/index.php?title=MAC%20adresa&oldid=6385716

Príloha A

Obsah CD

- **doc/** - adresár obsahujúci programovú dokumentáciu vygenerovanú nástrojom pdoc3
- **examples/** - adresár obsahujúci vymyslené sady filtrovacích pravidiel a formáty potrebné pre ich spracovanie v implementovanom nástroji
- **filter_rule_analyzer/** - adresár obsahujúci zdrojové kódy implementovaného nástroja
- **tex/** - adresár obsahujúci zdrojové kódy pre vytvorenie pdf súboru s technickou správou
- **README** - textový súbor obsahujúci manuál pre spustenie implementovaného nástroja
- **xsaboj00-Analyza-parametru-pravidlovych-sad.pdf** - pdf súbor s technickou správou práce

Príloha B

Príklad parametrového súboru formátu nástroja ClassBench

```
-scale
733
#
-prots
0 0.08458390 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
1 0.03137790 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
6 0.87312412 0.21562500 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.13124999 0.00000000 0.00000000 0.00000000
0.65312499 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
17 0.01091405 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.12500000 0.00000000 0.00000000 0.00000000
0.87500000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
#
-flags
0 0x0000/0x0000,1.00000000
1 0x0000/0x0000,1.00000000
6 0x0000/0x0000,0.81250000 0x0000/0x0200,0.09375000 0x1000/0x1000,0.09375000
17 0x0000/0x0000,1.00000000
#
-extra
0
#
-spar
#
```

```
-spem
#
-dpar
0.08235294 1600:1649
0.07058824 1300:1349
0.07058824 1300:1350
0.04705882 7500:7599
0.04705882 2200:2210
0.03529412 1025:65535
0.03529412 1700:1750
0.03529412 5001:65535
0.03529412 32200:32207
0.03529412 8000:8100
0.02352941 61700:61709
0.02352941 61500:61509
0.02352941 61600:61609
0.02352941 61900:61909
0.02352941 62000:62009
0.02352941 61800:61809
0.02352941 61100:61119
0.02352941 61200:61209
0.02352941 61000:61009
0.02352941 61400:61409
0.02352941 61300:61309
0.02352941 62700:62709
0.02352941 1700:1720
0.02352941 62600:62609
0.02352941 20000:20010
0.02352941 1600:1650
0.02352941 62400:62409
0.02352941 62100:62109
0.02352941 62500:62509
0.01176471 62200:62219
0.01176471 32210:32219
0.01176471 62220:62239
0.01176471 62200:62239
0.01176471 1300:1399
#
-dpem
0.18823530 1521:1521
0.05411765 1526:1526
0.04235294 1221:1221
0.03529412 1433:1433
0.02823529 1525:1525
0.02823529 135:135
0.02823529 14753:14753
0.02352941 1712:1712
0.02352941 1711:1711
```

```
0.02352941 1704:1704
0.02352941 1708:1708
0.02117647 5555:5555
0.02117647 1705:1705
0.02117647 6000:6000
0.02117647 20:20
0.02117647 21:21
0.02117647 1724:1724
0.02117647 1717:1717
0.01882353 1707:1707
0.01882353 1733:1733
0.01882353 1706:1706
0.01882353 5540:5540
0.01882353 5541:5541
0.01647059 1734:1734
0.01411765 1489:1489
0.01411765 1490:1490
0.01411765 5632:5632
0.01411765 5631:5631
0.01176471 6789:6789
0.00941176 1715:1715
0.00941176 2121:2121
0.00705882 6790:6790
0.00705882 32200:32200
0.00705882 32201:32201
0.00470588 2110:2110
0.00470588 2115:2115
0.00470588 3031:3031
0.00470588 1540:1540
0.00470588 31000:31000
0.00470588 3745:3745
0.00470588 30210:30210
0.00470588 19856:19856
0.00470588 2026:2026
0.00470588 443:443
0.00470588 80:80
0.00470588 30211:30211
0.00470588 6849:6849
0.00470588 27000:27000
0.00470588 27400:27400
0.00470588 65500:65500
0.00470588 30804:30804
0.00470588 30899:30899
0.00235294 1650:1650
0.00235294 1600:1600
0.00235294 15126:15126
0.00235294 15121:15121
0.00235294 2051:2051
```

```
0.00235294 2308:2308
0.00235294 12050:12050
0.00235294 1601:1601
0.00235294 1602:1602
0.00235294 5543:5543
0.00235294 7205:7205
0.00235294 5542:5542
0.00235294 1432:1432
0.00235294 1352:1352
0.00235294 6888:6888
0.00235294 512:512
0.00235294 4646:4646
0.00235294 55444:55444
0.00235294 6889:6889
0.00235294 1550:1550
0.00235294 6890:6890
#
-wc_wc
0,0.00448431 0,1.00000000
8,0.00896861 0,0.50000000 8,0.50000000
13,0.00448431 0,1.00000000
23,0.00448431 23,1.00000000
24,0.00448431 23,1.00000000
25,0.00448431 23,1.00000000
26,0.00448431 23,1.00000000
27,0.00448431 23,1.00000000
28,0.00448431 23,1.00000000
29,0.00448431 23,1.00000000
30,0.00448431 23,1.00000000
31,0.00448431 23,1.00000000
38,0.00448431 23,1.00000000
39,0.05381166 23,1.00000000
47,0.00448431 23,1.00000000
48,0.00448431 23,1.00000000
49,0.00448431 23,1.00000000
52,0.01793722 30,1.00000000
53,0.00896861 31,1.00000000
54,0.08968610 23,0.10000000 32,0.89999998
55,0.10313901 23,0.95652175 31,0.04347826
56,0.11210762 32,1.00000000
62,0.01793722 30,1.00000000
63,0.03139013 31,0.85714287 32,0.14285715
64,0.48878923 32,1.00000000
#
-wc_hi
#
-hi_wc
#
```

```
-hi_hi
#
-wc_lo
#
-lo_wc
#
-hi_lo
#
-lo_hi
#
-lo_lo
#
-wc_ar
55,0.01176471 23,1.00000000
63,0.03529412 31,1.00000000
64,0.95294118 32,1.00000000
#
-ar_wc
#
-hi_ar
#
-ar_hi
#
-wc_em
51,0.00470588 23,1.00000000
52,0.01411765 30,1.00000000
53,0.00705882 31,1.00000000
54,0.04470588 23,0.05263158 32,0.94736844
55,0.02823529 23,1.00000000
62,0.01176471 32,1.00000000
63,0.04235294 31,0.61111110 32,0.38888890
64,0.84705883 32,1.00000000
#
-em_wc
#
-hi_em
#
-em_hi
#
-lo_ar
#
-ar_lo
#
-lo_em
#
-em_lo
#
-ar_ar
```

```
#
-ar_em
#
-em_ar
#
-em_em
#
-snest
4
#
-sskew
0 0.00000000 1.00000000 0.99862826
1 1.00000000 0.00000000 1.00000000
2 1.00000000 0.00000000 1.00000000
3 1.00000000 0.00000000 1.00000000
4 1.00000000 0.00000000 1.00000000
5 1.00000000 0.00000000 1.00000000
6 0.50000000 0.50000000 0.99862635
7 1.00000000 0.00000000 1.00000000
8 1.00000000 0.00000000 1.00000000
9 1.00000000 0.00000000 1.00000000
10 1.00000000 0.00000000 1.00000000
11 1.00000000 0.00000000 1.00000000
12 1.00000000 0.00000000 1.00000000
13 1.00000000 0.00000000 1.00000000
14 1.00000000 0.00000000 1.00000000
15 1.00000000 0.00000000 1.00000000
16 1.00000000 0.00000000 1.00000000
17 1.00000000 0.00000000 1.00000000
18 1.00000000 0.00000000 1.00000000
19 1.00000000 0.00000000 1.00000000
20 1.00000000 0.00000000 1.00000000
21 1.00000000 0.00000000 1.00000000
22 1.00000000 0.00000000 1.00000000
23 1.00000000 0.00000000 1.00000000
24 0.50000000 0.50000000 0.49318182
25 0.33333334 0.66666669 0.73978359
26 0.20000000 0.80000001 0.32311377
27 0.22222222 0.77777779 0.36449289
28 0.25000000 0.75000000 0.46052003
29 0.25000000 0.75000000 0.36467236
30 0.46808508 0.53191489 0.48157272
31 0.73913044 0.26086956 0.46088934
32 0.00000000 1.00000000 0.46088934
#
-dnest
4
#
```

```
-dskew
0 0.00000000 1.00000000 0.73702419
1 0.00000000 1.00000000 0.61089814
2 0.33333334 0.66666669 0.79079866
3 0.25000000 0.75000000 0.87861329
4 0.16666667 0.83333337 0.65362722
5 0.60000002 0.40000001 0.83289701
6 0.69230765 0.30769232 0.58234501
7 0.76470590 0.23529412 0.49952471
8 0.80000001 0.20000000 0.77720296
9 0.91666669 0.08333334 0.95143843
10 0.92307693 0.07692308 0.69306612
11 0.82142860 0.17857143 0.53058130
12 0.75757575 0.24242425 0.53885406
13 0.77499998 0.22499999 0.58464622
14 0.81632656 0.18367347 0.63084495
15 0.85964912 0.14035088 0.71374631
16 0.90566039 0.09433962 0.92839336
17 0.89655173 0.10344828 0.62802786
18 0.95312500 0.04687500 0.53703707
19 0.92537314 0.07462686 0.70964915
20 0.95833331 0.04166667 0.63632482
21 0.90666670 0.09333333 0.52626264
22 0.96249998 0.03750000 0.48550725
23 0.90361446 0.09638554 0.31080827
24 0.93023258 0.06976745 0.40086997
25 0.91208792 0.08791209 0.58988094
26 0.91836733 0.08163265 0.61624527
27 0.93396229 0.06603774 0.50176704
28 0.90178573 0.09821429 0.34686840
29 0.86178863 0.13821138 0.46549407
30 0.89928055 0.10071942 0.31395534
31 0.90604025 0.09395973 0.47606516
32 0.00000000 1.00000000 0.47606516
#
-pcorr
1 0.20879121
2 0.43708611
3 1.00000000
4 0.87878788
5 0.86206895
6 0.83999997
7 1.00000000
8 1.00000000
9 0.97619045
10 1.00000000
11 1.00000000
12 1.00000000
```

```
13 1.00000000
14 1.00000000
15 1.00000000
16 1.00000000
17 1.00000000
18 1.00000000
19 1.00000000
20 1.00000000
21 1.00000000
22 0.00000000
23 0.00000000
24 0.00000000
25 0.00000000
26 0.00000000
27 0.00000000
28 0.00000000
29 0.00000000
30 0.00000000
31 0.00000000
32 0.00000000
#
```

Príloha C

Príklad parametrového súboru formátu nástroja ClassBench-ng

```
-scale
13778
#
-prots
0 0.3103498330672086 0.9679607109448082 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.032039289055191766
17 0.2551894324285092 0.6968145620022753 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.3031854379977247 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
6 0.25243141239657424 0.7044278320874066 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.29557216791259344 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1 0.18202932210770795 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#
-flags
0 0x0000/0x0000,1.00000000
17 0x0000/0x0000,1.00000000
6 0x0000/0x0000,1.00000000
1 0x0000/0x0000,1.00000000
#
-extra
0
#
-spar
#
-spem
0.9197080291970803 67:67
0.0364963503649635 2888:2888
0.0072992700729927005 443:443
0.0364963503649635 22:22
#
-dpar
#
```

```
-dpem
0.05199462124607799 8192:8192
0.05199462124607799 1:1
0.05199462124607799 2048:2048
0.005378753922008068 61440:61440
0.005378753922008068 57344:57344
0.05199462124607799 4096:4096
0.05199462124607799 256:256
0.005378753922008068 65024:65024
0.05199462124607799 4:4
0.05199462124607799 64:64
0.005378753922008068 65408:65408
0.05199462124607799 8:8
0.005378753922008068 65520:65520
0.05199462124607799 16384:16384
0.03182429403854774 67:67
0.005378753922008068 65532:65532
0.05199462124607799 2:2
0.05199462124607799 32768:32768
0.05199462124607799 16:16
0.05199462124607799 1024:1024
0.005378753922008068 63488:63488
0.04885701479157328 68:68
0.05199462124607799 32:32
0.005378753922008068 65528:65528
0.005378753922008068 65472:65472
0.005378753922008068 49152:49152
0.05199462124607799 128:128
0.005378753922008068 65280:65280
0.05199462124607799 512:512
0.005378753922008068 64512:64512
0.005378753922008068 65504:65504
0.0017929179740026895 443:443
0.0017929179740026895 8443:8443
0.005378753922008068 65534:65534
0.0017929179740026895 22:22
0.0017929179740026895 3389:3389
0.00044822949350067237 45518:45518
0.00044822949350067237 45530:45530
0.00044822949350067237 40502:40502
0.00044822949350067237 45527:45527
0.00044822949350067237 45521:45521
0.00044822949350067237 35008:35008
0.00044822949350067237 45524:45524
0.00044822949350067237 38356:38356
0.00044822949350067237 57929:57929
0.00044822949350067237 38352:38352
0.00044822949350067237 38365:38365
```

```
#
-wc_wc
64,0.9994803845154585 32,1.0
40,8.660258075690655e-05 32,1.0
56,8.660258075690655e-05 24,1.0
55,8.660258075690655e-05 23,1.0
62,8.660258075690655e-05 30,1.0
36,8.660258075690655e-05 4,1.0
52,8.660258075690655e-05 20,1.0
#
-wc_hi
#
-hi_wc
#
-hi_hi
#
-wc_lo
#
-lo_wc
#
-hi_lo
#
-lo_hi
#
-lo_lo
#
-wc_ar
#
-ar_wc
#
-hi_ar
#
-ar_hi
#
-wc_em
64,1.0 32,1.0
#
-em_wc
#
-hi_em
#
-em_hi
#
-lo_ar
#
-ar_lo
#
-lo_em
```

```
#
-em_lo
#
-ar_ar
#
-ar_em
#
-em_ar
#
-em_em
64,1.0 32,1.0
#
-snest
1
#
-sskew
0 0.0 1.0 0.9545500251382605
1 0.0 1.0 0.5359929339025468
2 0.0 1.0 0.9880907752041772
3 0.75 0.25 0.9747003994673769
4 0.8 0.2 0.6956144159791576
5 0.8333333333333334 0.16666666666666666 0.9832229580573951
6 0.7142857142857143 0.2857142857142857 0.6842970166652276
7 0.7777777777777778 0.2222222222222222 0.7461365099806825
8 0.9 0.1 0.9967700258397932
9 0.8181818181818182 0.18181818181818182 0.9690445516809689
10 1.0 0.0 0.0
11 1.0 0.0 0.0
12 0.9230769230769231 0.07692307692307693 0.7176470588235294
13 0.8571428571428571 0.14285714285714285 0.6678082191780822
14 1.0 0.0 0.0
15 1.0 0.0 0.0
16 0.9375 0.0625 0.9976069398743643
17 1.0 0.0 0.0
18 0.8235294117647058 0.17647058823529413 0.49452782989368355
19 0.85 0.15 0.49063710872296934
20 0.9565217391304348 0.043478260869565216 0.7818181818181819
21 0.7916666666666666 0.20833333333333334 0.748101963356065
22 0.7586206896551724 0.2413793103448276 0.6276331456287173
23 0.6111111111111112 0.3888888888888889 0.49384178996279193
24 0.62 0.38 0.3075823236773248
25 0.5757575757575758 0.42424242424242425 0.23646709280295494
26 0.5425531914893617 0.4574468085106383 0.18149615429139418
27 0.49635036496350365 0.5036496350364964 0.13074395855860385
28 0.2807017543859649 0.7192982456140351 0.38317258552895755
29 0.35051546391752575 0.6494845360824743 0.3448024522585926
30 0.49375 0.50625 0.22833426861204636
31 0.6569037656903766 0.34309623430962344 0.04181184668989549
```

```
32 0.0 0.0 0.0
#
-dnest
2
#
-dskew
0 0.0 1.0 0.027447698744769822
1 0.0 1.0 0.26004659289458354
2 0.5 0.5 0.9992122883024813
3 0.3333333333333333 0.6666666666666666 0.48807394156231365
4 0.75 0.25 0.950891460744448
5 0.6 0.4 0.9697815500620608
6 0.8571428571428571 0.14285714285714285 0.9450511945392491
7 0.875 0.125 0.10588235294117643
8 0.8888888888888888 0.1111111111111111 0.9375
9 0.8 0.2 0.9489707434776052
10 1.0 0.0 0.0
11 1.0 0.0 0.0
12 1.0 0.0 0.0
13 0.9166666666666666 0.08333333333333333 0.012987012987012991
14 1.0 0.0 0.0
15 1.0 0.0 0.0
16 0.9230769230769231 0.07692307692307693 0.7102803738317758
17 0.8571428571428571 0.14285714285714285 0.43175287356321834
18 0.625 0.375 0.4813902218940979
19 0.6818181818181818 0.3181818181818182 0.47424064275916133
20 0.6071428571428571 0.39285714285714285 0.4404761904761905
21 0.7692307692307693 0.23076923076923078 0.5677248677248677
22 0.8958333333333334 0.10416666666666667 0.6551219512195121
23 0.7924528301886793 0.20754716981132076 0.4103077091547137
24 0.8125 0.1875 0.38967230944842884
25 0.8947368421052632 0.10526315789473684 0.2707473635946813
26 0.8809523809523809 0.11904761904761904 0.2
27 0.9148936170212766 0.0851063829787234 0.23967470760233917
28 1.0 0.0 0.0
29 1.0 0.0 0.0
30 0.9285714285714286 0.07142857142857142 0.5
31 1.0 0.0 0.0
32 0.0 0.0 0.0
#
-pcorr
1 0.0
2 0.0
3 0.0
4 0.0
5 0.0
6 0.0
7 0.0
```

```
8 0.0
9 0.0
10 0.0
11 0.0
12 0.0
13 0.0
14 0.0
15 0.0
16 0.0
17 0.0
18 0.0
19 0.0
20 0.0
21 0.0
22 0.0
23 0.0
24 0.0
25 0.0
26 0.0
27 0.0
28 0.0
29 0.0
30 0.0
31 0.0
32 0.0
#
-openflow
--
in_port:
'71': 2
'50': 2
'393': 2
'471': 2
'249': 2
'308': 2
'180': 2
'522': 2
'81': 2
'69': 2
'70': 2
'360': 2
'1': 2
'269': 2
'173': 2
'470': 2
'310': 2
'289': 2
'521': 2
```

```
'37554': 1
'37618': 1
'37742': 1
'37653': 1
'37553': 1
'37677': 1
'37612': 1
'37702': 1
'37715': 1
'37607': 1
'37660': 1
'37598': 1
'37670': 1
'37703': 1
'37640': 1
'37672': 1
'37696': 1
'37597': 1
'36736': 1
'37737': 1
'37621': 1
'37225': 1
'37690': 1
'37562': 1
'37680': 1
'37617': 1
'36697': 1
'37555': 1
'37716': 1
'37611': 1
'37561': 1
'37581': 1
'37663': 1
'37714': 1
'36666': 1
'36464': 1
'37606': 1
'37681': 1
'37605': 1
'37748': 1
'37616': 1
'36662': 1
'37725': 1
'37664': 1
'37700': 1
'36717': 1
'37548': 1
'37719': 1
```

```
'36803': 1
'37710': 1
'37673': 1
'37676': 1
'37587': 1
'37651': 1
'37659': 1
'37685': 1
'37602': 1
'37579': 1
'37600': 1
'37620': 1
'37614': 1
'37695': 1
'37648': 1
'37549': 1
'37736': 1
'37705': 1
'37678': 1
'37662': 1
'37622': 1
'37665': 1
'37642': 1
'37720': 1
'37722': 1
'37655': 1
'36646': 1
'37713': 1
'37744': 1
'37647': 1
'37686': 1
'37727': 1
'37563': 1
'37738': 1
'37080': 1
'37550': 1
'37552': 1
'37675': 1
'37583': 1
'36361': 1
'37551': 1
'37646': 1
'37656': 1
'36658': 1
'37545': 1
'37601': 1
'37643': 1
'37657': 1
```

```
'37560': 1
'37084': 1
'37741': 1
'37743': 1
'37585': 1
'37556': 1
'37683': 1
'37688': 1
eth_type:
'0x800': 9502
dl_src:
fa:16:3e: 838
dl_dst:
01:80:c2: 1
01:00:0c: 4
00:e0:2b: 3
fa:16:3e: 12534
ff:ff:ff: 6
'01:00:00': 85
c2:81:09: 15
'00:00:00': 2
unique_vlan_ids_count: 0
empty_rules_count: 3190
rule_distribution:
- attributes:
- dl_dst
count: 1016
- attributes:
- in_port
count: 142
- attributes:
- dl_src
- eth_type
- nw_proto
- nw_src
- tp_dst
count: 384
- attributes:
- dl_dst
- nw_dst
count: 180
- attributes:
- dl_dst
- eth_type
- nw_dst
- nw_proto
- tp_dst
count: 1710
```

```
- attributes:
- nw_src
count: 3
- attributes:
- nw_dst
- tp_dst
- tp_src
count: 71
- attributes:
- dl_dst
- eth_type
- nw_dst
- nw_proto
- nw_src
count: 7224
- attributes:
- nw_dst
count: 81
- attributes:
- dl_dst
- nw_dst
- tp_dst
- tp_src
count: 54
- attributes:
- dl_dst
- eth_type
- nw_dst
- nw_proto
count: 46
- attributes:
- dl_dst
- nw_dst
- nw_src
count: 2408
- attributes:
- dl_src
count: 81
- attributes:
- dl_src
- nw_src
count: 228
- attributes:
- dl_src
- eth_type
- nw_proto
- nw_src
count: 138
```

```
- attributes:
- dl_dst
- dl_src
- nw_dst
- nw_src
- tp_dst
- tp_src
count: 7
- attributes:
- dl_dst
- nw_dst
- nw_src
- tp_dst
- tp_src
count: 5
#
```