VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ALGORITMY PRO KLASIFIKACI PAKETŮ

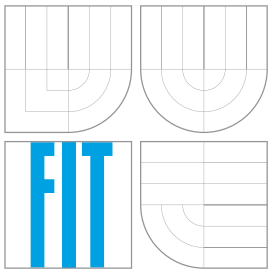
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL MYŠKA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ALGORITMY PRO KLASIFIKACI PAKETŮ

ALGORITHMS FOR PACKET CLASSIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL MYŠKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. VIKTOR PUŠ
SUPERVISOR

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá problematikou klasifikace paketů v počítačových sítích. V úvodu je vysvětlena teorie popisující počítačové sítě a přístupy ke klasifikaci. Dále je vybrán klasifikační algoritmus založený na tvorbě rozhodovacího stromu. Důležitá vlastnost tohoto algoritmu je možnost parametrizace. V práci je analyzován vliv jednotlivých parametrů na tvorbu rozhodovacího stromu. Vybraný klasifikační algoritmus je také porovnán s jinými klasifikačními algoritmy z experimentální knihovny Netbench.

Abstract

This bachelor's thesis deals with packet classification in computer networks. The theory describing a computer network and approach to the classification is explained in the introduction. Then, classification algorithm based on making decision tree is chosen. An important feature of this algorithm is the possibility of parameterization. The work analyzes the influence of various parameters on the creation of a decision tree. Selected classification algorithm is also compared with other classification algorithms from experimental library Netbench.

Klíčová slova

Klasifikace paketů, filtrování paketů, HiCuts, klasifikační algoritmus

Keywords

Packet classification, packet filters, HiCuts, classification algorithm

Citace

Michal Myška: Algoritmy pro klasifikaci paketů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Algoritmy pro klasifikaci paketů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Viktora Puše.

.....
Michal Myška
9. května 2012

Poděkování

Děkuji panu Ing. Viktoru Pušovi za ochotu a pomoc při tvorbě této bakalářské práce.

Obsah

1 Úvod **3**

2 Teorie počítačových sítí **4**
2.1 ISO/OSI model . . . 4
2.2 TCP/IP model . . . 5
2.3 Bezestavový firewall . . . 5
2.4 Stavový firewall . . . 6
2.5 Aplikační brány . . . 6
2.6 Klasifikační pravidla . . . 6
2.7 Klasifikace paketů . . . 8
2.8 Metriky klasifikačních algoritmů . . . 8

3 Klasifikační algoritmy **9**
3.1 DCFL . . . 10
3.2 MSCA . . . 12
3.2.1 Bloomův filtr . . . 14
3.3 PHCA . . . 14

4 HiCuts **16**
4.1 Datová struktura . . . 16
4.2 Heuristiky algoritmu . . . 17
4.2.1 Heuristika pro výběr počtu řezů . . . 17
4.2.2 Heuristika pro výběr dimenze . . . 18
4.2.3 Heuristika znovupoužití uzlů . . . 19
4.2.4 Heuristika odstranění redundance . . . 19

5 Implementace **20**
5.1 Netbench . . . 20
5.2 Sestavení rozhodovacího stromu . . . 20
5.3 Nalezení optimální dimenze . . . 21
5.4 Heuristika znovupoužití uzlů . . . 21
5.5 Klasifikace . . . 21
5.6 Metriky . . . 23

6 Experimenty **24**
6.1 Snížení spotřeby paměti při použití heuristiky znovupoužití uzlů . . . 24
6.2 Porovnání vlastností algoritmu, při různých parametrech . . . 24
6.3 Porovnání spotřebované paměti různých algoritmů . . . 25

7 Závěr **29**

A Obsah CD **32**

Kapitola 1

Úvod

Internet v dnešní době představuje rozsáhlou síť, využívanou velkým počtem uživatelů a podporující množství různorodých aplikací. Tento stav vyžaduje implementaci spousty nových mechanizmů zajišťujících bezpečnost a různou kvalitu služeb.

Jeden ze základních bezpečnostních prvků je firewall. [12] Je to zařízení, které provádí kontrolu paketů a na základě definovaných pravidel blokuje nebo propouští dané pakety.

Počítačové sítě podporují různé přenosy dat, proto je potřeba rozlišit jednotlivé typy provozu a zacházet s nimi podle jejich potřeb. K tomu slouží systém zajišťující kvalitu služeb (QoS, Quality of Service). [6] Systém zajišťující kvalitu služeb je soubor mechanizmů pro efektivní využití dané šířky pásma tak, aby se co nejvíce vyhovělo všem aplikacím, využívajícím síťové prostředky. Mezi další požadavky na zajištění kvality služeb patří také spolehlivost dané služby a určení maximálního zpoždění dat. Systém pro zajištění kvality služeb je možné postavit na dvou základních modelech. První model představuje integrované služby. Ty klasifikují jednotlivé sítově toky do definovaných tříd a pro každý síťový tok rezervují síťové zdroje. Druhý model představuje diferenciované služby. Ty jsou založeny na klasifikaci a značení paketů na hraničních směrovačích a přeposílání paketů ve vnitřní síti, dle definovaných politik.

Všechny výše zmíněné mechanizmy využívají klasifikaci paketů. Pro rychlost dnešních počítačových sítí je výpočetní výkon personálních počítačů nedostačující, obzvláště rychlost operačních pamětí. Z tohoto důvodu jsou kladeny vysoké nároky na rychlost klasifikace. Proto je potřeba efektivních klasifikačních metod. Původně byly klasifikační metody řešeny softwarově. Dnes je snaha o tvorbu klasifikačních algoritmů, které je možné implementovat na hardware. Hardwarově řešené klasifikační algoritmy jsou rychlé, ale mají velké náklady na pozdější úpravy a optimalizace. U softwarově řešených algoritmů je jednoduchá pozdější úprava a optimalizace, ovšem jejich rychlost je pro dnešní sítě nedostatečná. [13]

V následující kapitole této práce jsou popsány základní vrstvové modely počítačových sítí. Dále je v ní uvedena problematika klasifikačních pravidel a jejich způsobu zápisu. Také je zde vysvětlen základní princip klasifikace paketů a základní metriky, podle kterých se klasifikační algoritmy hodnotí. Ve třetí kapitole jsou popsány vybrané klasifikační algoritmy z experimentální knihovny Netbench. Je zde také uvedeno rozdělení klasifikačních algoritmů do tříd. Ve čtvrté kapitole je popsán klasifikační algoritmus HiCuts. Také je zde vysvětlena datová struktura rozhodovacího stromu a heuristiky, které se využívají pro optimalizaci tvorby rozhodovacího stromu. V páté kapitole je popsána implementace algoritmu, se zaměřením na implementované heuristiky a průběh klasifikace. V šesté kapitole jsou popsány výsledky experimentů s klasifikačním algoritmem.

Kapitola 2

Teorie počítačových sítí

2.1 ISO/OSI model

Model ISO/OSI je referenční model pro popis síťové architektury, definovaný mezinárodní organizací OSI. Mnoho používaných protokolů má vytvořenou strukturu právě podle tohoto modelu. V praxi nikdy nebyl celý model implementován. Model ISO/OSI obsahuje sedm vrstev a každá vrstva popisuje protokoly a funkce pro přenos dat. Vrstvy využívají při komunikaci služby nižších vrstev. [14] [5]

- Fyzická vrstva

  Vrstva definující fyzické vlastnosti zařízení, jako jsou úrovně napětí pro logické kódování nebo fyzickou rychlost přenosového média. Hlavními funkcemi této vrstvy jsou navazování a ukončování spojení, efektivní rozložení zdrojů mezi uživatele, konverze digitálních dat na signály, používané přenosovým médiem. Používané datové jednotky na této vrstvě jsou bity.

- Linková vrstva

  Popisuje přenos dat po konkrétní datové lince. Datové jednotky na této vrstvě se nazývají rámce. Poskytuje spojení pouze mezi místně připojenými zařízeními. Na této vrstvě jsou použity například protokoly Ethernet, PPP.

- Síťová vrstva

  Vrstva zajišťující adresování a směrování dat. Poskytuje funkce k zajištění přenosu dat mezi koncovými zařízeními, které spolu přímo nesousedí. Protokoly pracující na této vrstvě jsou například IP, ICMP nebo ARP. Datovým jednotkám na této vrstvě se říká diagramy.

- Transportní vrstva

  Vrstva zajišťující přenos dat mezi koncovými uzly. Na této vrstvě pracují TCP a UDP protokoly. TCP je spojovaný přenosový protokol zajišťující spolehlivý přenos dat. UDP je nespojovaný přenosový protokol zajišťující přenos dat bez kontroly doručení. Datové jednotky na této vrstvě se nazývají pakety.

- Relační vrstva

  Vrstva sloužící k udržování spojení mezi dvěma aplikacemi. Zajišťuje vytvoření spojení, ukončení spojeni, synchronizaci.

- Prezentační vrstva

  Vrstva zajišťující zobrazení přenášených dat mezi různými aplikacemi v daném formátu. Zajišťuje také kompresi a kódování dat.

- Aplikační vrstva

  Nejvyšší vrstva definující aplikace komunikující po síti. Na této vrstvě pracují například protokoly SSH, DNS, FTP.

2.2 TCP/IP model

Oproti ISO/OSI modelu je TCP/IP model jednodušší a odstraňuje jeho nedostatky způsobené komplikovanou strukturou. Tento model je v dnešní době velmi rozšířený a používaný. TCP/IP model se skládá ze čtyř vrstev. Komunikace probíhá vždy mezi stejnými vrstvami a při komunikaci jsou využívány služby nižší vrstvy. Ilustrace zmíněných modelů je na obrázku 2.1. [5]

- Vrstva fyzického rozhraní

  Vrstva popisující standardy pro fyzické médium a signály. Poskytuje přístup k fyzickému přenosovému médiu. Také zajišťuje zapouzdření IP datagramů do rámců. Na této vrstvě pracuji například protokoly Ethernet, Token Ring.

- Síťová vrstva

  Datové jednotky této vrstvy se nazývají datagramy. Vrstva především zajišťuje adresování a směrování. K tomu je využíván protokol IP. Další protokoly pracující na této vrstvě jsou ICMP, IGMP, ARP.

- Transportní vrstva

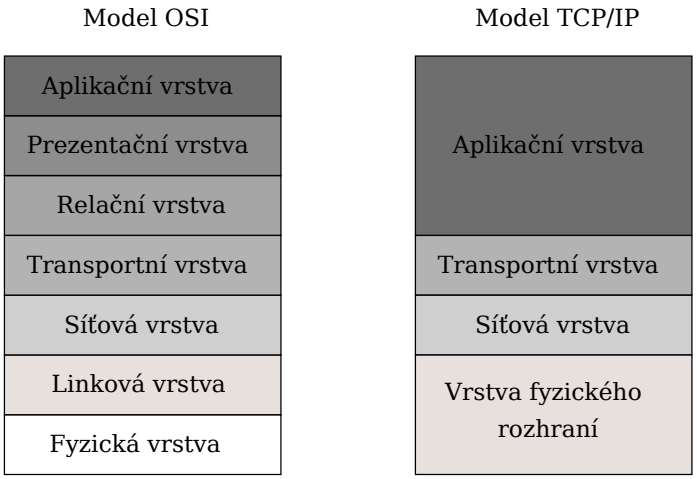
  Transportní vrstva zajišťuje přenos dat mezi koncovými zařízeními. Na této vrstvě pracují dva přenosové protokoly. TCP protokol umožňuje spojovaný přenos, s možností kontroly doručených dat. UDP protokol se používá u nespojovaných služeb, u kterých je potřeba rychlý přenos bez nutnosti spolehlivého doručení. Datové jednotky na této vrstvě se nazývají pakety.

- Aplikační vrstva

  Je tvořena aplikacemi a procesy, které spolu komunikují. Zajišťuje také reprezentaci a kódování dat. Protokoly této vrstvy můžeme rozdělit na uživatelské protokoly, jejichž služby využívá uživatel a systémové protokoly, které poskytují síťové funkce. Uživatelské protokoly mohou být například FTP, Telnet. Systémové protokoly mohou být například DNS, SNMP.

2.3 Bezestavový firewall

Bezestavový firewall [12] patří k nejjednodušším typům firewallu. Lze ho označit také jako paketový filtr. Jedná se o filtrování na základě hodnot v hlavičce paketu. Tyto hodnoty jsou porovnávány s definovanými pravidly a na jejich základě se provede s paketem patřičná operace.



Obrázek 2.1: Porovnání modelů OSI a TCP/IP. Zdroj [5]

Výhodou tohoto typu firewallu je vysoká rychlost zpracování. Nevýhodou je nízká úroveň kontroly, zejména u složitějších protokolů. Důvodem je, že bezestavový firewall kontroluje každý paket samostatně, bez jeho kontextu.

2.4 Stavový firewall

Stavový firewall [12] provádí kromě filtrování jednotlivých paketů na základě hodnot v hlavičce také ukládání informací o povolených spojeních. Tyto informace využívá k rozhodování, zda příchozí paket patří do skupiny povolených spojení a propustí ho bez kontroly nebo provede jeho klasifikaci. To zajišťuje rychlejší zpracování paketů a zároveň umožňuje tvorbu jednodušších pravidel.

K výhodám tohoto typu firewallu patří vysoká rychlost zpracování a vyšší úroveň zabezpečení, ve srovnání s bezestavovými firewally. Stavové firewally mají obecně nižší úroveň zabezpečení, než tomu je u aplikačních bran, zmíněných v další podkapitole.

2.5 Aplikační brány

Aplikační brány [12] na rozdíl od předchozích typů firewallu zcela oddělují okolní sítě od sítě, na kterou byly použity. Někdy se aplikační brány označují jako proxy firewally. Princip tohoto typu firewallu je založen na vytváření spojení vnějších sítí s aplikačními bránami. Aplikační brány zpracují spojení a vytvoří nová spojení s vnitřní sítí.

Výhodou tohoto typu firewallu je vysoká úroveň zabezpečení. Hlavní nevýhody jsou vysoká náročnost a podpora malého počtu protokolů. Dnes se aplikační brány téměř nepoužívají.

2.6 Klasifikační pravidla

Klasifikace paketů se provádí prohledáváním množiny pravidel. [8] Každé pravidlo se skládá z dvojice *(F, A)*. *A* je označení pro akci, provedenou při klasifikaci paketu, dle daného pravidla. Mezi typické akce patří zahození paketu, propuštění paketu nebo přesměrování.

F je množina podmínek, která je rozdělena na dimenze. Obecně může existovat jedna až N-dimenzí, podle kterých se klasifikují příchozí pakety. V této bakalářské práci jsou použita pětidimenzionální klasifikační pravidla. Použité dimenze jsou: zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port, protokol.

- IP adresa

  Slouží k identifikaci počítače v počítačové síti. Pracuje na síťové vrstvě modelu ISO/OSI. U IP protokolu verze 4 (IPv4) má tato adresa 32 bitů. U IP protokolu verze 6 má IP adresa 128 bitů. V této práci jsou použity pouze 32bitové IP adresy.

- Port

  Představuje 16bitové číslo, identifikující službu, která komunikuje v počítačové síti. Pracuje na transportní vrstvě modelu ISO/OSI.

- Protokol

  Při klasifikaci také kontrolujeme, jaký přenosový protokol byl u paketu použit. Typicky se jedna o protokoly TCP a UDP, pracující na transportní vrstvě. [5]

Hodnoty jednotlivých dimenzí mohou být zapsány několika způsoby:

- Rozsah

  Je zadán intervalem mezi největší a nejmenší zadanou hodnotou. Tento způsob zápisu se používá zejména u zdrojového portu a cílového portu.

- Prefix

  Prefix je zapsán pomocí dvojice, adresa a maska. Adresa představuje unikátní identifikaci zařízení v síti. Maska představuje počet vrchních bitů, použity pro adresaci sítě, ve které se zařízení nachází. Logickým součinem těchto hodnot lze získat adresu sítě. Typicky se používá tento způsob zápisu u zdrojové IP adresy a cílové IP adresy.

- Hodnota

  Dimenze zadaná hodnotou je klasifikována na přesnou shodu. Tento zápis se používá u dimenze protokol.

V některých případech se může stát, že pro jeden paket, může být nalezeno více pravidel. Proto každé pravidlo obsahuje také hodnotu, která určuje prioritu pravidla, a také je použita pro indexaci pravidla v celé množině. Tato hodnota je zadána jako přirozené číslo a musí být vzhledem k množině pravidel unikátní.

Níže je uveden příklad klasifikačního pravidla. V tabulce 2.1 jsou vysvětlené jeho jednotlivé položky.

$$1264\ pass\ proto\ tcp\ from\ 147.251.48.28/16\ to\ 147.251.5.48\ port\ 8000:8120$$

Část pravidla	Vysvětlení
1264	číslo a priorita pravidla
pass	akce provedená s paketem
proto tcp	použitý protokol paketu
from 147.251.48.28/16	zdrojová IP adresa se zadaným prefixem
to 147.251.5.48	cílová IP adresa
port 8000:8120	cílový port zadaný rozsahem hodnot

Tabulka 2.1: Vysvětlení jednotlivých částí výše zmíněného pravidla.

2.7 Klasifikace paketů

Klasifikace paketů je proces rozpoznání příchozích paketů, na základě množiny pravidel. Na základě identifikace nejlépe odpovídajícího pravidla se provede příslušná akce s paketem. Klasifikace porovnává jedna až N různých dimenzí z hlavičky paketu, viz podkapitola 2.6. Vyhledávat v dimenzích můžeme několika způsoby. Vyhledáním nejdelšího shodného prefixu, vyhledáním rozsahu hodnot, vyhledáním přesné shody hodnot.

Hlavní problémy při řešení klasifikace paketů jsou vysoké nároky na výkon. Je to dáno vysokorychlostním přenosem dat dnešních počítačových sítí. Proto byly zavedeny metriky, podle kterých se klasifikační algoritmy hodnotí. [3]

2.8 Metriky klasifikačních algoritmů

- Rychlost vyhledávání

  Klasifikační algoritmy musí být dostatečně rychlé, aby nezdržovaly procesy, které je využívají. Rychlost se odvozuje od nejhoršího možného případu. Například linka s rychlostí 10 Gb/s může přenést v jednom směru až $19,5$ miliónů paketů za sekundu. [9]

- Paměťové požadavky

  Spotřeba paměti má vliv také na rychlost algoritmu. Paměti s menší velikostí mají vyšší rychlost než paměti s větší velikostí. U softwarově řešených klasifikačních algoritmů je možné, pří nízké spotřebě paměti, vložit klasifikátor do rychlejších vyrovnávacích pamětí. U hardwarově řešených klasifikačních algoritmů je možné, při nízké spotřebě paměti, využít interních pamětí. Spotřeba paměti také ovlivňuje cenu výroby klasifikátoru.

- Rychlost aktualizace

  Při změnách v množině pravidel, jako přidání nebo odebrání daného pravidla, je potřeba, aby se této změně přizpůsobila struktura klasifikátoru. Pro různá zařízení jsou různé nároky na rychlost změny. Například u firewallů, kde se pravidla vkládají nebo odebírají ručně, nejsou tak velké nároky na rychlost aktualizace. Ve směrovačích jsou však nároky na rychlost větší.

Kapitola 3

Klasifikační algoritmy

Klasifikační algoritmy lze rozdělit do několika tříd, podle techniky vyhledávání: [1]

- Vyhledávání hrubou silou

  Jedná se o jeden ze základních typů vyhledávání. Postupně se prohledává celá množina pravidel, dokud se nenalezne odpovídající pravidlo. Dá se rozdělit na dva přístupy. Lineární vyhledávání a ternární varianta obsahem adresované paměti (TCAM). Lineární vyhledávání představuje nejméně výkonnou alternativu, zatímco TCAM představuje nejvýkonnější klasifikační algoritmus.

- Vyhledávání založené na dekompozici

  Dekompoziční algoritmy prohledávají každou dimenzi samostatně. Výsledky jednotlivých operací poté shromáždí a podle nich se vyhledá příslušné pravidlo. Hlavní problém tohoto vyhledávání je najít efektivní metodu pro zpracování shromážděných výsledků. Výhodou naopak je možnost paralelního vyhledávání v jednotlivých dimenzích.

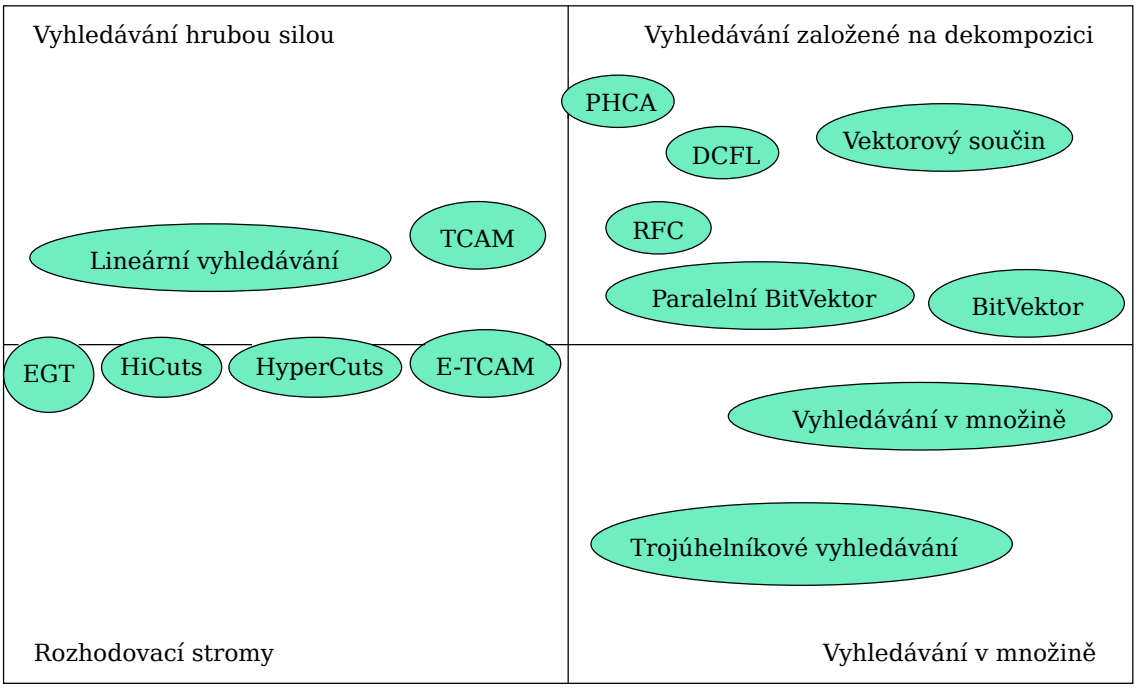
- Vyhledávání založené na rozhodovacích stromech

  Nejdříve se během předzpracování vytváří rozhodovací strom. Vyhledávání poté představuje průchod vytvořeným rozhodovacím stromem. Rozhodovací stromy obsahují zpravidla vnitřní uzly a listy. Vnitřní uzly obsahují informace pro další průchod a listy obsahují číselné označení konkrétních pravidel. Při tvorbě rozhodovacího stromu může nastat komplikace, že pravidla můžeme porovnávat různými způsoby, jako vyhledání nejdelšího shodného prefixu nebo porovnání na přesnou shodu. Proto je dobré převést na jeden typ porovnávání.

- Vyhledávání v množině

  Základním principem je definování počtu platných bitů jednotlivých dimenzí. Tento počet máme uložený v množině a ta nám slouží jako vyhledávací klíč. Na základě tohoto klíče provádíme vyhledávání na přesnou shodu jen určených bitů jednotlivých dimenzí. Prohledávání jednotlivých složek množiny je nezávislé, a proto je možné jej provádět paralelně.

Ilustrace rozdělení algoritmů do tříd je na obrázku 3.1. V dalších podkapitolách je uvedeno několik klasifikačních algoritmů. Tyto algoritmy jsou k nalezení v experimentální knihovně Netbench.



Obrázek 3.1: Roztřídění klasifikačních algoritmů podle techniky vyhledávání. Zdroj [1]

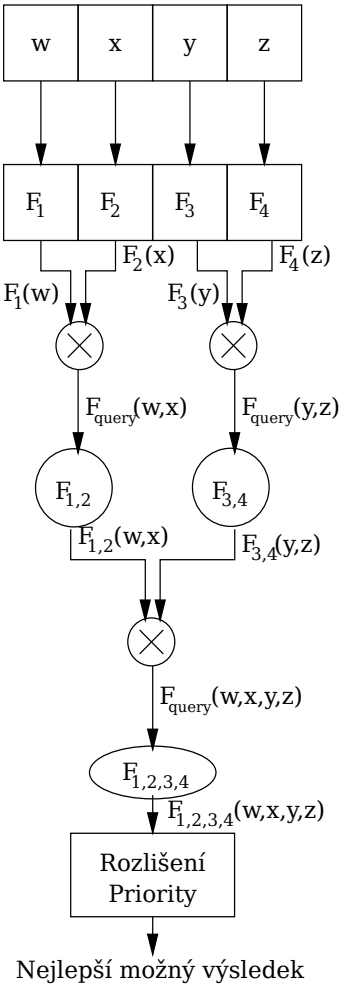
3.1 DCFL

Algoritmus Distributed Crossproducting of Field Labels [2] je klasifikační algoritmus představující kombinaci starých a nových klasifikačních technik, využívající funkce hardwarového paralelismu. V optimálním případě představuje paralelismus vyhledávání pro každou dimenzi zvlášť.

Algoritmus vychází ze dvou klíčových faktů. Počet unikátních hodnot pro danou dimenzi v pravidlech je relativně málo, vzhledem k celkovému počtu pravidel. Počet unikátních hodnot pro danou dimenzi odpovídající danému paketu je velmi málo, vzhledem k celkovému počtu pravidel. Funkcionalita algoritmu může být strukturována různým způsoby. Jeden ze způsobu je vysvětlen na obrázku 3.2.

Předpokládejme čtyřdimenzionální vyhledávání. Hodnoty jednotlivých dimenzí příchozích paketů představují proměnné w, x, y, z. F_1, F_2, F_3, F_4 jsou množiny unikátních hodnot jednotlivých dimenzí. Příchodem hodnot paketů jsou z množin unikátních hodnot paralelně nalezeny podmnožiny $F_1(w)$, $F_2(x)$, $F_3(y)$, $F_4(z)$. Poté jsou paralelně vytvořeny podmnožiny $F_{1,2}(w, x)$, $F_{3,4}(y, z)$. Před jejich vytvořením je nejdřív vytvořena podmnožina $F_{query}(w, x)$, vektorovým součinem podmnožin $F_1(w)$ a $F_2(x)$ a podmnožina $F_query(y, z)$, vektorovým součinem podmnožin $F_3(y)$ a $F_4(z)$. Poté je každá hodnota z podmnožiny F_{query}(w, x) hledána v množině $F_{1,2}$. Pokud je nalezena, je přidána do podmnožiny $F_{1,2}(w, x)$. Vše probíhá symetricky při vytvoření podmnožiny $F_{3,4}(y, z)$.

V dalším kroku je nalezena podmnožina $F_{1,2,3,4}(w, x, y, z)$, shodou hodnot množiny $F_{1,2,3,4}$ a podmnožiny $F_{query}(w, x, y, z)$, která byla vytvořena vektorovým součinem podmnožin $F_{1,2}(w, x)$ a $F_{3,4}(y, z)$. Z podmnožiny $F_{1,2,3,4}(w, x, y, z)$ je dle priorit vybráno jedno nebo více pravidel, nejlépe odpovídající danému paketu.



Obrázek 3.2: Schéma klasifikace algoritmu Distributed Crossproducting of Field Labels. [2]

3.2 MSCA

Multi-Subset Crossproduct Algorithm [7] je klasifikační algoritmus využívající Bloomův filtr, který má vysokou paměťovou efektivitu. Rozděluje množinu pravidel na několik podmnožin. Tento počin má za snahu odstranit pseudopravidla z jednotlivých podmnožin. Tím se radikálně sníží režie. Pro optimální výběr počtu podmnožin se používá heuristika. Pokud bude menší počet podmnožin, bude velký počet pseudopravidel. Při velkém počtu podmnožin bude růst spotřeba paměti, velkým množstvím Bloomových filtrů. Pseudopravidla jsou vyžadována tehdy, když pravidla v jedné podmnožině mají překrývající prefixy. Proto heuristika vytváří podmnožiny, které neobsahují překrývající se pravidla.

Algoritmus využívá vyhledávání nejdelšího shodného prefixu. Jsou vytvořeny tabulky prefixů pro každou podmnožinu. Jelikož se předem neví, ve které tabulce je nejdelší prefix, musí se prohledávat tabulky jednotlivých podmnožin. To zvyšuje počet přístupů do paměti a snižuje efektivitu algoritmu.

Aby se tomu předešlo, vytváří se globální tabulka prefixů, obsahující unikátní prefixy pro každou dimenzi ze všech podmnožin. Poté je použit Bloomův filtr k vyhledání daného pravidla. Popis Bloomova filtru je v podkapitole 3.2.1.

Celý proces klasifikace vyžaduje $4 + p$ přístupů do paměti. Počet je dán čtyřmi kontrolovanými dimenzemi a proměnnou p, která znamená počet hashovacích funkcí. Multi-Subset Crossproduct Algorithm klasifikuje pouze v těchto čtyřech dimenzích: zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port.

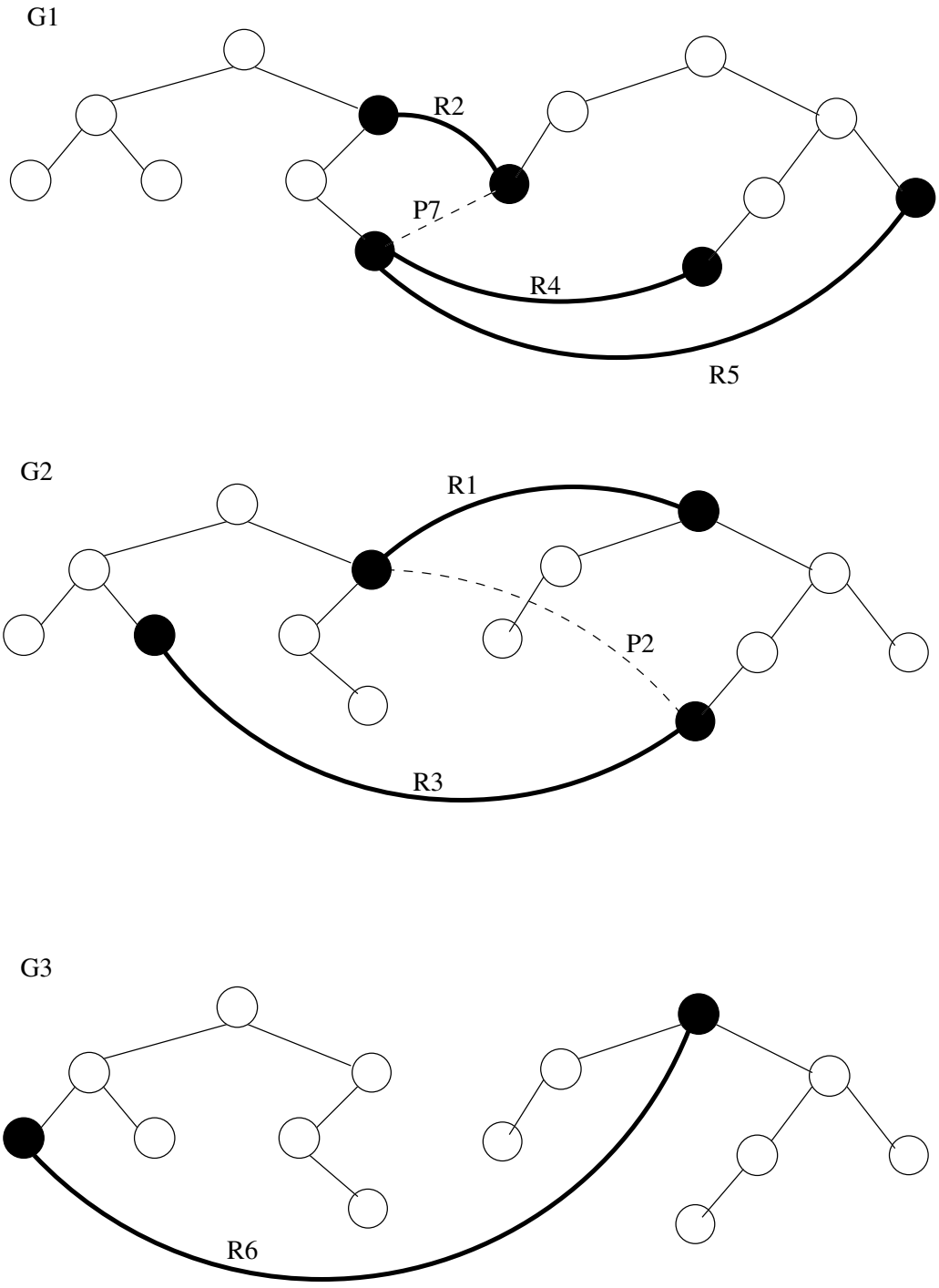
Na obrázku 3.3 je ukázán příklad tohoto klasifikačního algoritmu. Struktura je složena ze třech podmnožin $G1, G2, G3$. V tabulce 3.1 je množina pravidel. Pro každou dimenzi jsou vytvořeny tabulky nejdelšího shodného prefixu (tabulka 3.2 a tabulka 3.3).

Pravidlo	Dimenze 1	Dimenze 2
R1	1*	*
R2	1*	00*
R3	01*	100*
R4	101*	100*
R5	101*	11*
R6	00*	*

Tabulka 3.1: Tabulka pravidel, obsahující dvě dimenze. [7]

Dimenze 1	G1	G2	G3
1*	1	1	
00*			2
01*		2	
101*	3	1	

Tabulka 3.2: Tabulka nejdelšího shodného prefixu pro *Dimenzi 1.* [7]



Obrázek 3.3: Schéma struktury Multi-Subset Crossproduct Algorithm. [7]

Dimenze 2	G1	G2	G3
*		0	0
00*	2	0	0
11*	2	0	0
100*	3	3	0

Tabulka 3.3: Tabulka nejdelšího shodného prefixu pro *Dimenzi 2*. [7]

3.2.1 Bloomův filtr

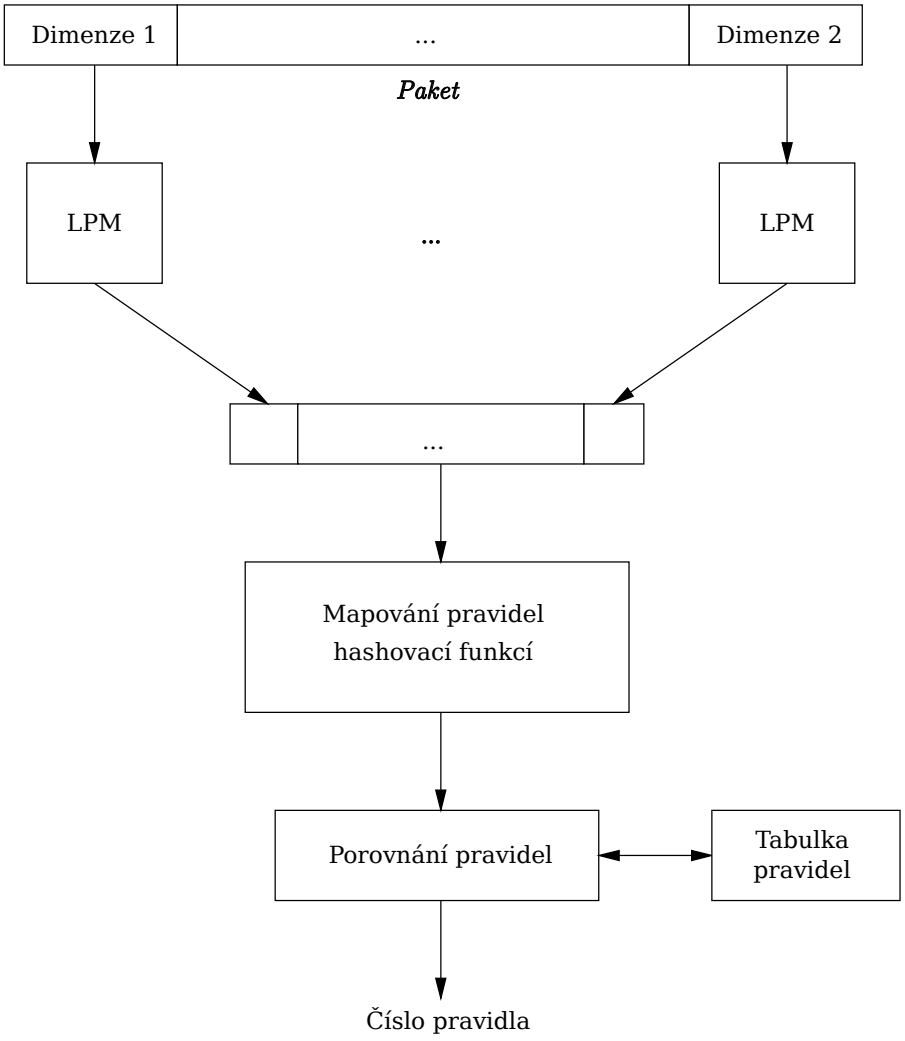
Bloomův filtr je pravděpodobnostní datová struktura, sloužící k ověřování příslušnosti prvku k množině. Vyžaduje mít definovaný určitý počet hashovacích funkcí, přiřazené všem možným prvkům, u kterých budeme chtít zjišťovat příslušnost k množině. Při ověřování může nastat chyba. Ověřovaný prvek může byt označen jako patřící do množiny, i když ve skutečnosti do ní nepatří. Pravděpodobnost výskytu chyby roste s počtem prvků v množině. [11]

3.3 PHCA

Perfect Hash Crossproduct Algorithm [9] [10] je založen na dekompozici. To přináší řadu výhod. Jednou z nich je možnost paralelního zpracování. PHCA je velmi rychlý algoritmus, jeho nevýhodou však zůstává velká paměťová náročnost. Tento algoritmus si udržuje konstantní složitost pro přístup do paměti. Pro klasifikaci jednoho paketu jsou potřeba pouze dva přístupy do paměti. Struktura tohoto klasifikačního algoritmu je ilustrována na obrázku 3.4

Prvním krokem je vyhledání nejdelšího shodného prefixu (LPM) pro každou dimenzi z hlavičky paketu. Rozsahy jednotlivých dimenzí jsou převedeny na prefixy. Jednotlivým prefixům jsou přiřazená čísla. Poté se provede konkatenace jednotlivých prefixů a vznikne jedno datové slovo.

Datové slovo může odpovídat nějakému pravidlu nebo pseudopravidlu. V dalším kroku se provede perfektní hashovací funkce. Je použito statické hashovaní, protože dynamické má daleko větší režii. Při konstrukci se vytváří acyklický graf, jehož hrany jsou klíče a jeho uzly jsou výsledky dvou rozdílných hashovacích funkcí. Zde jsou klíče pravidla a pseudopravidla ve formě konkatenace výsledků nejdelšího shodného prefixu. Nad nimi se provedou obě hashovací funkce. Výsledky těchto funkci jsou požity jako adresa do tabulky vrcholů, ze které se zjistí jejich ohodnocení. Po sečtení ohodnocení získáme právě jedno číslo pravidla, odpovídající danému paketu. Pokud paketu neodpovídá ani jedno pravidlo, výsledek je náhodné číslo pravidla. V posledním kroku klasifikace zkontroluje pomocí tabulky pravidel, zda skutečně pravidlo odpovídá danému paketu.



Obrázek 3.4: Schéma klasifikace Perfect Hash Crossproduct Algorithm. [17]

Kapitola 4

HiCuts

Hierarchical Intelligent Cuttings (HiCuts) [4] je algoritmus pro vícedimenzionální klasifikaci paketů. Základní koncept vychází z geometrického pohledu na klasifikaci. Datová struktura je založena na konstrukci rozhodovacího stromu, na základě přijaté množiny pravidel. Rozhodovací strom se skládá z vnitřních uzlů, obsahující potřebné informace pro klasifikaci, a listů, obsahující číselné označení daných pravidel. Rozhodovací strom je vytvořen během předzpracování na základě vstupních pravidel pro klasifikaci. Využívá kombinaci stromového vyhledávání, doplněnou o lineární vyhledávání v listech. Lineární vyhledávání bylo zavedeno kvůli úspoře paměti. Samotná klasifikace spočívá v průchodu rozhodovacího stromu od kořenu až k listům, kde jsou uložena čísla příslušných pravidel. Tento algoritmus má snahu o rovnováhu mezi rychlostí klasifikace a paměťovou náročností.

4.1 Datová struktura

Jak již bylo zmíněno, HiCuts vychází z konstrukce rozhodovacího stromu, obsahující vnitřní uzly a listy. Typické vlastnosti rozhodovacího stromu jsou hloubka, počet vnitřních uzlů, počet listů. Při tvorbě rozhodovacího stromu, ke každému vnitřnímu uzlu přiřazujeme:

- Interval z celkového rozsahu dané dimenze
- Daný řez, určený dimenzí a počtem řezů
- Podmnožinu pravidel, která odpovídá danému intervalu

Na obrázku 4.2 je příklad rozhodovacího stromu. Slouží ke klasifikaci ve dvou dimenzích, s rozsahem hodnot na čtyřech bitech. Pravidla pro sestavení rozhodovacího stromu jsou v tabulce 4.1. Pro sestavení rozhodovacího stromu lze převést množinu pravidel do geometrické reprezentace, znázorněnou na obrázku 4.1.

Spotřeba paměti rozhodovacího stromu je odvozena struktury vnitřních uzlů, struktury listů a odkazů na vnitřní uzly a listy.

- Vnitřní uzly

  Vnitřní uzly obsahují informaci o aktuální dimenzi. V podkapitole 2.6 je zmíněno, že počet možných dimenzí je pět. Tato informace se dá uložit na třech bitech.

  Dále se ukládá informace o počtu řezů. V této práci byl maximální počet řezů omezen na 128. Tuto informaci dokážeme uložit na sedmi bitech.

- Listy

  V listech se ukládají indexy pravidel o maximálním počtu daným parametrem *binth*. Největší množina pravidel obsahovala 1265 pravidel. Z toho byla určena potřebná paměť pro uložení jednoho pravidla, která je jedenáct bitů.

- Adresa paměti

  Pro tuto práci byla určena adresa paměti na 16 bitů. Adresy do paměti využíváme při průchodu rozhodovacím stromem. Každý vnitřní uzel obsahuje adresy svých potomků. Jelikož vnitřní uzel může mít pouze jediný otcovský uzel, je použitý vždy jeden odkaz do paměti na jeden vnitřní uzel.

  S použitím heuristiky znovupoužití uzlů, která je popsána v podkapitolách 4.2.3 a 5.4, může nastat, že na jeden list existuje více odkazů. Proto pro výpočet paměti bude potřeba znát počet odkazů, ne jen počet listů rozhodovacího stromu.

Vzorec pro výpočet celkové spotřeby paměti rozhodovacího stromu:

$$paměť = 26 * počet\ vnitřních\ uzlů + 11 * počet\ pravidel + 16 * počet\ odkazů\ na\ listy$$

Pravidlo	Adresa	Port
a	1010	2:2
b	1100	5:5
c	0101	8:8
d	*	6:6
e	111*	0:15
f	001*	9:15
g	00*	0:4
h	0*	0:3
i	0110	0:15
j	1*	7:15
k	0*	11:11

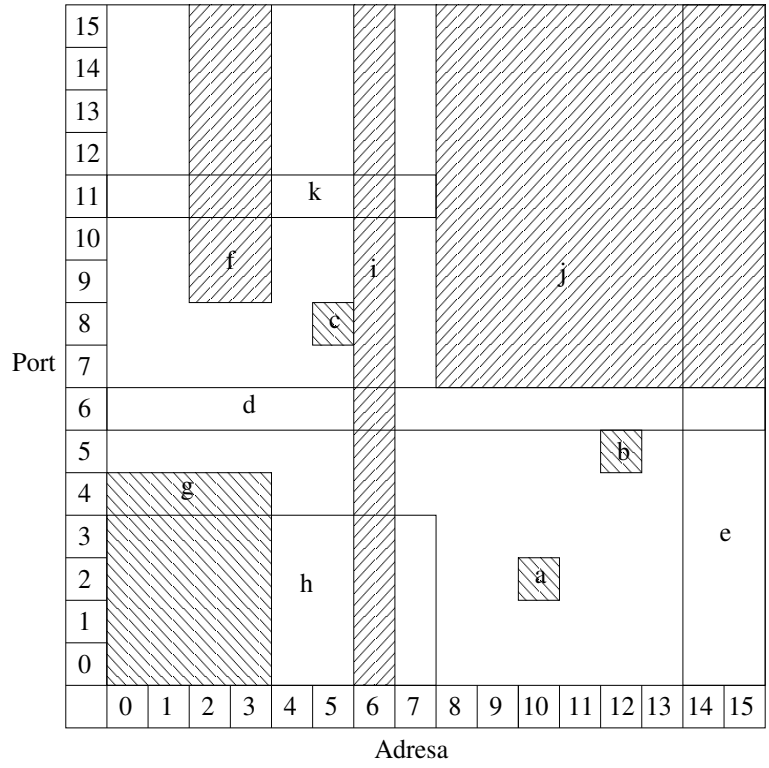
Tabulka 4.1: Množina pravidel obsahující dvě dimenze, adresu a port. [1]

4.2 Heuristiky algoritmu

Existuje mnoho způsobů, jak sestavit rozhodovací strom. Během předzpracování HiCuts využívá několik heuristik v závislosti na struktuře klasifikátoru:

4.2.1 Heuristika pro výběr počtu řezů

Vybírá vhodný počet řezů v daném vnitřním uzlu. Proměnnou určující počet řezů označujeme *np*. Příliš velký počet řezů sníží počet úrovní stromu, což sníží dobu vyhledávání, ale zvýší paměťovou náročnost. Naopak malý počet řezů sníží paměťovou náročnost a zvýší dobu vyhledávání. K vyvážení rychlosti a paměťové náročnosti se využívá funkce $spmf()$,



Obrázek 4.1: Geometrická reprezentace rozhodovacího stromu, dle množiny pravidel na tabulky 4.1. Osy představují jednotlivé dimenze a pravidla představují obdélníky, úsečky nebo body. [1]

která je odvozena z n počtu pravidel pro daný uzel a konstantou *spfac*, získanou jako parametr. Vzorec pro výpočet *spmf()*:

$$spmf() = spfac * n$$

Také definujeme proměnnou *sm*, součtem pravidel všech potomků daného uzlu a aktuálního počtu řezů.

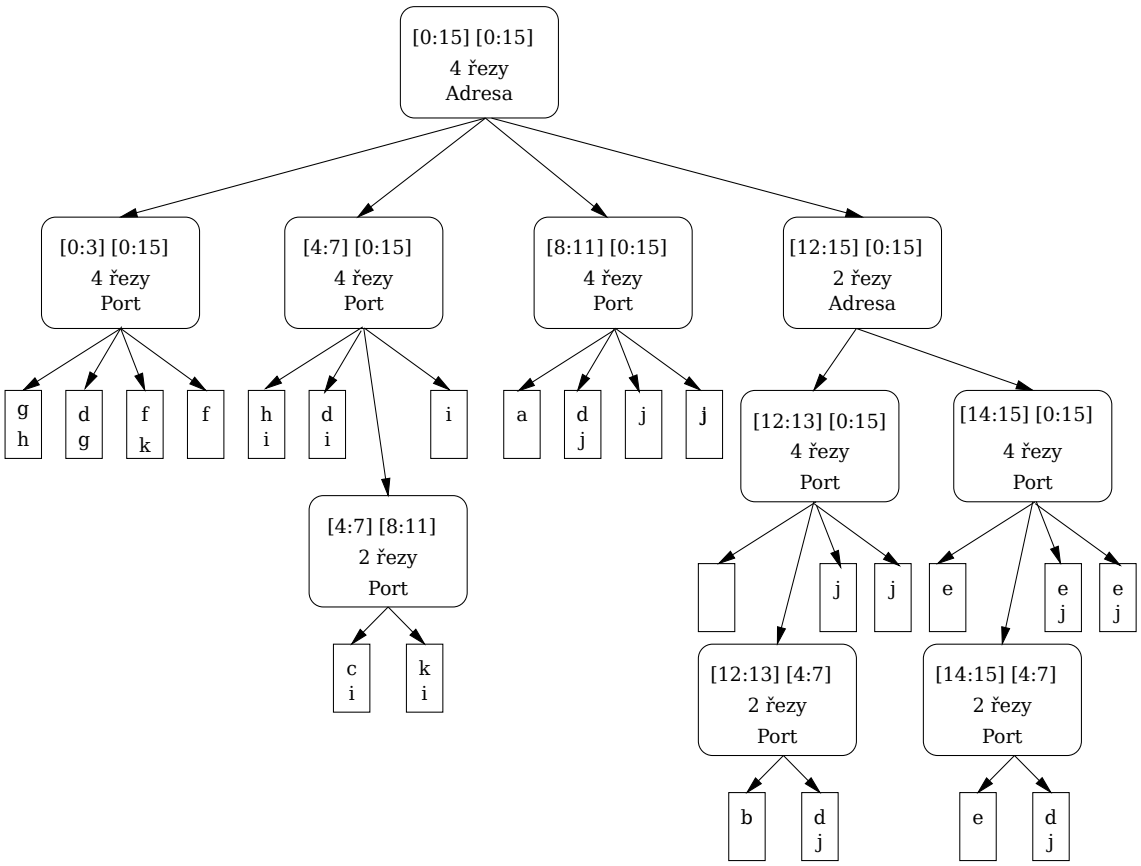
$$sm = \sum_{i=1}^{np} n\ pravidel\ v\ potomku + np$$

Poté optimální počet řezů daného uzlu získáme za podmínky, kdy $sm < spmf()$.

4.2.2 Heuristika pro výběr dimenze

Tato heuristika provádí výběr dimenze pro každý vnitřní uzel. Existuje několik způsobů, jak vybrat správnou dimenzi:

- Minimalizovat maximální počet pravidel všech potomků, při pokusu o snížení největšího počtu úrovní stromu.
- Vybrání takové dimenze, která vede k co nejvíce rovnoměrnému rozložení pravidel v uzlech. Při této heuristice lze dosáhnout vyváženého rozhodovacího stromu.



Obrázek 4.2: Podoba dvojdimenzionálního rozhodovacího stromu algoritmu HiCuts. Uzly obsahují označení a interval dané dimenze. Listy stromu obsahují označení pravidel odpovídající hodnotám dimenzí, při průchodu stromem. [1]

- Minimalizace proměnné *sm* ve všech dimenzích.
- Výběr dimenze s největším počtem různých položek pravidel.

4.2.3 Heuristika znovupoužití uzlů

Tato heuristika vede k co největšímu znovupoužití potomků. Pokud je přiřazena více potomkům stejná množina pravidel, potom můžou být tyto uzly sloučeny do jednoho.

4.2.4 Heuristika odstranění redundance

Slouží k odstranění redundance v rozhodovacím stromu. Při provádění řezu ve vnitřním uzlu může dojít k redundanci, protože některá pravidla v řezu můžou být překryta pravidlem s vyšší prioritou. Detekce redundance se používá v případě, pokud počet přiřazených pravidel daného uzlu je menší než maximální počet pravidel v listech stromu.

Kapitola 5

Implementace

Implementace HiCuts algoritmu byla provedena v programovacím jazyce Python, tak aby byl program kompatibilní s experimentální knihovnou Netbench. Program je rozdělen do několika částí. V první části dojde k sestavení rozhodovacího stromu na základě získané množiny pravidel. Detailnější popis sestavení rozhodovacího stromu je v podkapitole 5.2. V druhé části dochází ke klasifikaci. Klasifikační funkce dostane na vstup hlavičku paketu a po provedení vrátí odpovídající číslo pravidla. V třetí části program tiskne statistiky vytvořeného rozhodovacího stromu.

5.1 Netbench

Netbench [15] je experimentální knihovna obsahující různé algoritmy z oblasti zpracování paketů. Byla vytvořená výzkumnou skupinou *Akcelerovaných síťových technologií*. Jejím hlavním cílem je sloužit jako nezávislá platforma pro výzkumné pracovníky, při realizaci jejich algoritmů z oblasti zpracování paketů. Také slouží jako ukázka implementací různých algoritmů, s možností vyzkoušení jejich funkcionality. Knihovna je psána v programovacím jazyce Python. V dnešní době poskytuje implementační rozhraní pro experimenty z oblasti klasifikace paketů, vyhledání nejdelšího shodného prefixu, hledání řetězců.

5.2 Sestavení rozhodovacího stromu

Jak již bylo zmíněno v předcházející kapitole, sestavení lze ovlivnit několika parametry. Parametr *binth* určuje maximální možný počet pravidel uchovávaných v listech stromu. Parametr *spfac* ovlivňuje počet řezů v daném uzlu. Při sestavení rozhodovacího stromu musíme v každém uzlu uchovávat:

- Počet řezů

  Tento atribut určuje, kolik potomků bude aktuální uzel mít. K jeho určení byla použita heuristika, popsána v podkapitole 4.2.1, kde je označován jako proměnná *np*.

- Dimenzi

  Tento atribut určuje, ve které dimenzi budou prováděny řezy. K jeho určení byla použita heuristika, blíže popsaná v podkapitole 5.3.

Pro každý uzel musíme určit aktuální rozsahy hodnot jednotlivých dimenzí a množinu pravidel odpovídající těmto rozsahům. V této práci jsou rozsahy jednotně převedeny do formy prefixů.

Sestavení začíná od kořenového uzlu, kterému jsou přiřazeny plné rozsahy dimenzí a celá množina pravidel. Na základě těchto hodnot je kořenový uzel inicializován, určením počtu řezů a dimenze, ve které se budou řezy provádět. Poté je vytvořen odpovídající počet potomků. Potomkům jsou přiřazeny nové rozsahy a množiny pravidel. Poté se postupně provádí inicializace potomků.

Pokud je některému uzlu přiřazena množina pravidel s menším počtem prvků, než určený parametrem *binth*, stává se z něj list. V listu jsou uchovány indexy přiřazených pravidel. Poté se výpočet vrací do nadřazeného uzlu. Na obrázku 5.1 je zobrazeno schéma vytvoření rozhodovacího stromu.

5.3 Nalezení optimální dimenze

Při vytváření rozhodovacího stromu je potřeba v každém vnitřním uzlu rozhodnout, ve které dimenzi se budou provádět řezy. Jak již bylo zmíněno v kapitole 4, existuje několik možných heuristik pro výběr dimenze. V této práci byla použita heuristika pro výběr dimenze na základě největšího počtu různých hodnot v dané dimenzi. Pro každou dimenzi se počítá jen pravidlo, pro které platí:

- Má definovanou hodnotu.
- Nepokrývá celý rozsah dimenze.
- Doposud se hodnota neobjevila v jiném pravidle.

Pokud se vyskytují pravidla s nedefinovanou hodnotou nebo hodnotou pokrývající celý rozsah, počítá se jen první z těchto výskytů.

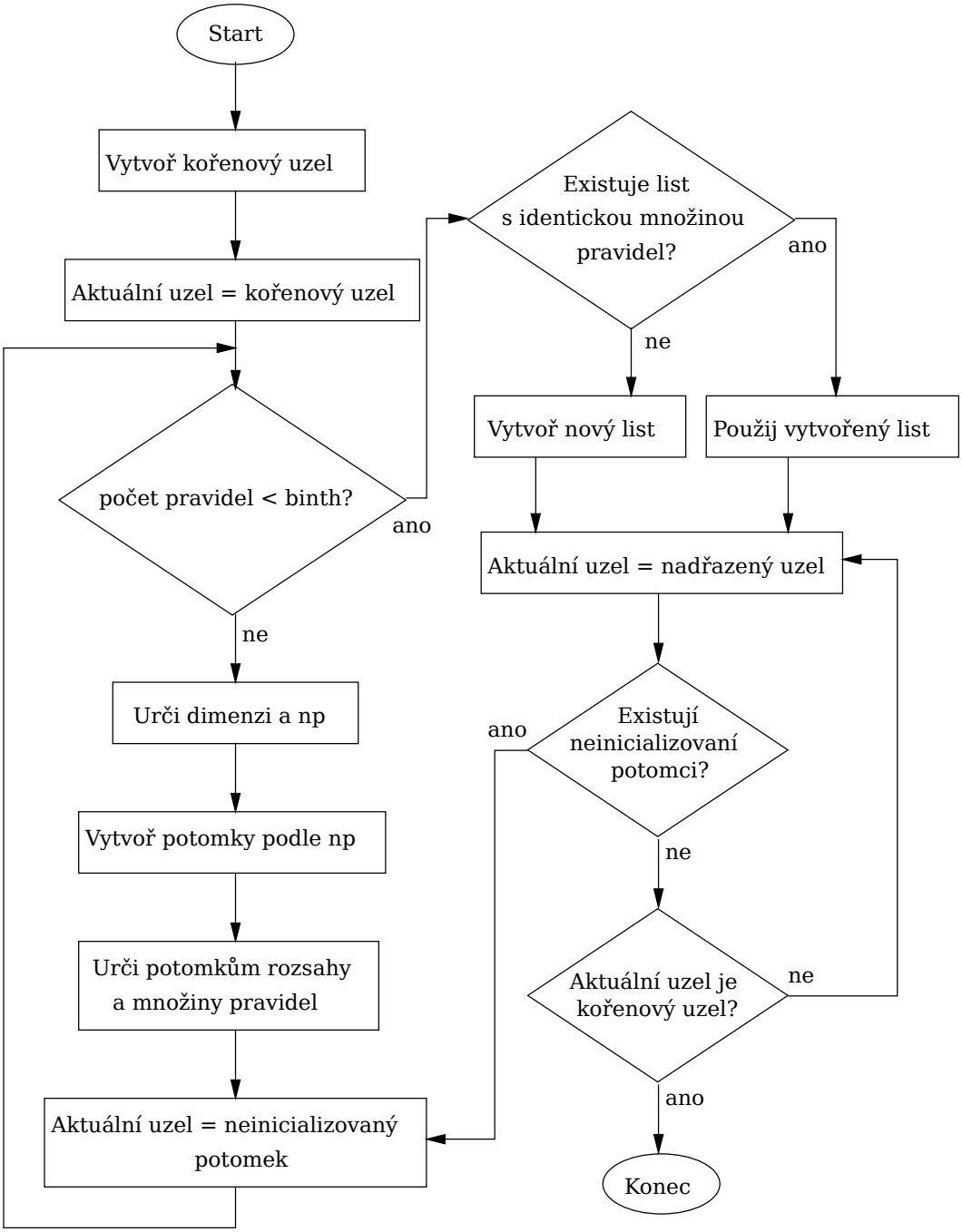
5.4 Heuristika znovupoužití uzlů

V podkapitole 4.2.3 byla zmíněná heuristika pro snížení paměťové náročnosti, znovupoužitím potomků, s přiřazenou identickou množinou pravidel. V této práci byla tato heuristika zpracována jiným způsobem. První změnou je znovupoužití pouze listů stromu. Další změnou je aplikace této heuristiky na listy celého stromu, nikoliv jen potomky daného uzlu.

Při tvorbě nového listu se kontroluje, zda množina pravidel tohoto listu je identická s množinou pravidel některého z již vytvořených listů. Pokud se žádná množina pravidel již vytvořených listů neshoduje s množinou pravidel aktuálního listu, daný list se vytvoří. V opačném případě se použije již existující list.

5.5 Klasifikace

Při klasifikaci dochází k průchodu rozhodovacího stromu. Kromě perzistentních hodnot počtu řezů a vybrané dimenze je potřeba také určit množinu pravidel a rozsah hodnot pro každou dimenzi, při průchodu vnitřními uzly. Klasifikace začíná průchodem od kořene stromu. U vnitřního uzlu je potřeba najít potomka s rozsahem hodnot v dané dimenzi,



Obrázek 5.1: Diagram vytvoření rozhodovacího stromu.

odpovídající danému paketu. Přitom dochází ke kontrole vrchních bitů, které jsou dané počtem řezů dle vzorce:

$$Počet\ bitů = log_2(počet\ řezů)$$

Hodnota určených bitů odpovídá pozici hledaného potomka v pořadí potomků aktuálního uzlu. Po nalezení potomka pokračuje prohledávání v další úrovni stromu. V listech stromu dochází k lineárnímu prohledávání pravidel, s kontrolou přesné shody.

5.6 Metriky

V tabulce 5.1 jsou vypsány základní metriky kódu klasifikačního algoritmu. Pro jejich získání byl použit program PyLint. [18]

Menší počet řádků implementovaného algoritmu je způsoben použitím již implementovaných tříd z experimentální knihovny Netbench. Byly použity třídy pro množiny pravidel, hlavičky paketů a práci s hodnotami ve formě prefixů.

Typ metriky	Hodnota
Počet řádků	570
Počet řádků komentářů	158
Počet prázdných řádků	61
Počet vytvořených tříd	2
Počet implementovaných metod	12

Tabulka 5.1: Základní metriky kódu implementovaného klasifikačního algoritmu.

Kapitola 6

Experimenty

V této kapitole jsou zhodnoceny výsledky experimentů s implementovaným klasifikačním algoritmem. Pracuje se s množinami reálných pravidel, kromě porovnání paměťových nároků různých klasifikačních algoritmů. Zde byly použity i množiny pseudopravidel.

V tabulce 6.1 jsou popsány vlastnosti použitých množin pravidel. Soubory *fw2_05_m05_250.rul* a *fw2_05_m05_500.rul* jsou množiny pseudopravidel z experimentální knihovny *Netbench*. Ostatní soubory jsou množiny reálných pravidel.

Množiny pravidel	Pravidla	Zdrojová IP adresa	Cílová IP adresa	Protokol	Zdrojový port	Cílový port
fw2_05_m05_250.rul	219	55	53	1	14	1
fw2_05_m05_500.rul	394	65	57	1	14	1
rules2.rul	173	84	84	3	1	16
rules3.rul	103	28	48	4	6	40
rules4.rul	275	46	64	3	1	22
rules6.rul	1 107	158	80	4	1	56

Tabulka 6.1: Základní vlastnosti použitých množin pravidel.

6.1 Snížení spotřeby paměti při použití heuristiky znovupoužití uzlů

V podkapitolách 5.4 a 4.2.3 je vysvětlen princip heuristiky znovupoužití uzlů. Byly provedeny experimenty pro různé parametry rozhodovacího stromu, s použitím této heuristiky i bez jejího použití. V tabulce 6.2 jsou uvedeny hodnoty spotřebované paměti. Na obrázku 6.1 je graf s porovnáním spotřeby paměti programu s použitím heuristiky a bez ní. Tento experiment byl proveden nad množinou pravidel *rules6.out*. Základní vlastnosti této množiny pravidel jsou uvedeny v tabulce 6.1.

6.2 Porovnání vlastností algoritmu, při různých parametrech

U parametrizovatelných programů je snaha najít takové parametry, které zajišťují nejlepší vlastnosti. K nalezení takových parametrů byla použita Paretova fronta. [16]

Binth/spfac	S heuristikou	Bez heuristiky
4/2	880, 68	992, 02
8/2	702, 02	791, 80
16/2	673, 96	756, 47
32/2	664, 57	743, 30
4/4	1622, 75	1811, 57
8/4	1091, 38	1263, 03
16/4	1038, 98	1190, 27
32/4	1018, 89	1158, 19
4/8	2506, 14	2787, 84
8/8	1121, 15	1323, 71
16/8	1044, 05	1205, 43
32/8	1020, 11	1162, 38
4/16	2714, 84	3094, 21
8/16	1133, 83	1374, 77
16/16	1045, 42	1213, 49
32/16	1020, 11	1162, 38

Tabulka 6.2: Snížení spotřeby paměti s heuristikou znovupožití uzlů (kbity).

Vlastnosti, pro které byly hledány optimální parametry, jsou hloubka rozhodovacího stromu a spotřebovaná paměť. Parametry, ovlivňující tvorbu rozhodovacího stromu jsou zmíněny v podkapitole 5.2. V tabulce 6.3 jsou uvedeny hodnoty spotřebované paměti a hloubky stromu pro různé hodnoty parametrů. Na obrázku 6.2 je graf Paretovy fronty. Optimální parametry byly hledány u množiny pravidel *rules6.out*. V tabulce 6.1 jsou popsány vlastnosti této množiny pravidel.

V tabulce a grafu nejsou zobrazeny všechny parametry. Kvůli podobným hodnotám některých parametrů došlo k překrývání v grafu 6.2. Z překrývajících se parametrů byly ponechány takové, které vykazovaly lepší vlastnosti.

Z grafu 6.2 lze usoudit, že pro množinu pravidel *rules6.out* jsou vhodné parametry *binth = 8, spfac = 8*. Kombinace těchto parametrů nabízí kompromis mezi spotřebou paměti a hloubkou stromu.

6.3 Porovnání spotřebované paměti různých algoritmů

Paměťová náročnost je jedna z důležitých charakteristik klasifikačních algoritmů. V tomto experimentu je porovnána paměťová náročnost klasifikačních algoritmů z experimentální knihovny Netbench a algoritmu HiCuts. Pro sestavení rozhodovacího stromu byly použity parametry *binth = 16, spfac = 8*.

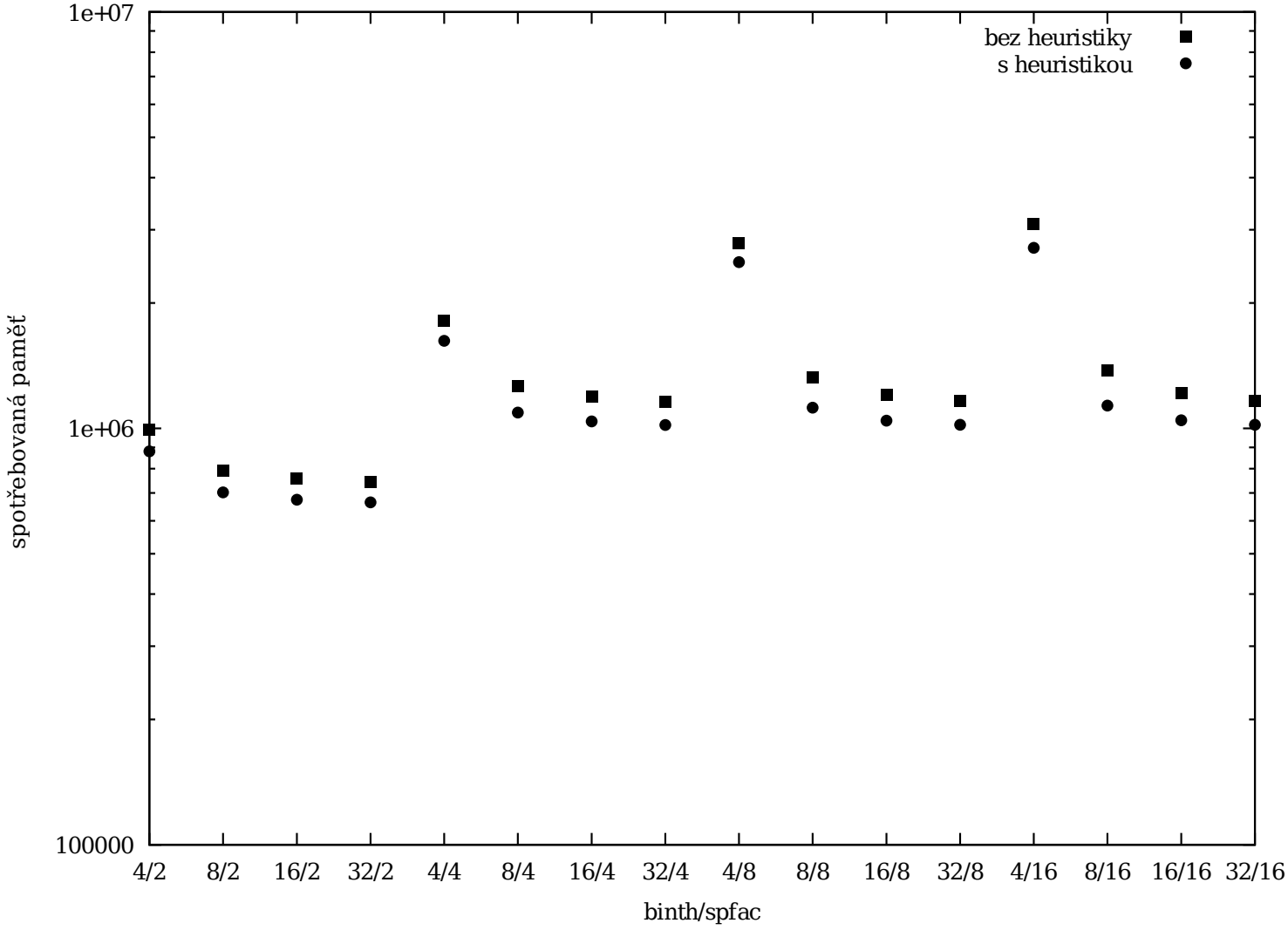
V tabulce 6.4 jsou uvedeny hodnoty spotřebované paměti pro vybrané množiny pravidel.

Binth/spfac	Paměť	Hloubka
4/2	880, 68	31
8/2	702, 02	23
16/2	673, 96	28
32/2	664, 57	43
4/4	1622, 75	19
8/4	1091, 38	21
16/4	1038, 98	27
32/4	1018, 89	42
4/8	2506, 14	17
8/8	1121, 15	20
16/8	1044, 05	26
4/16	2714, 84	17

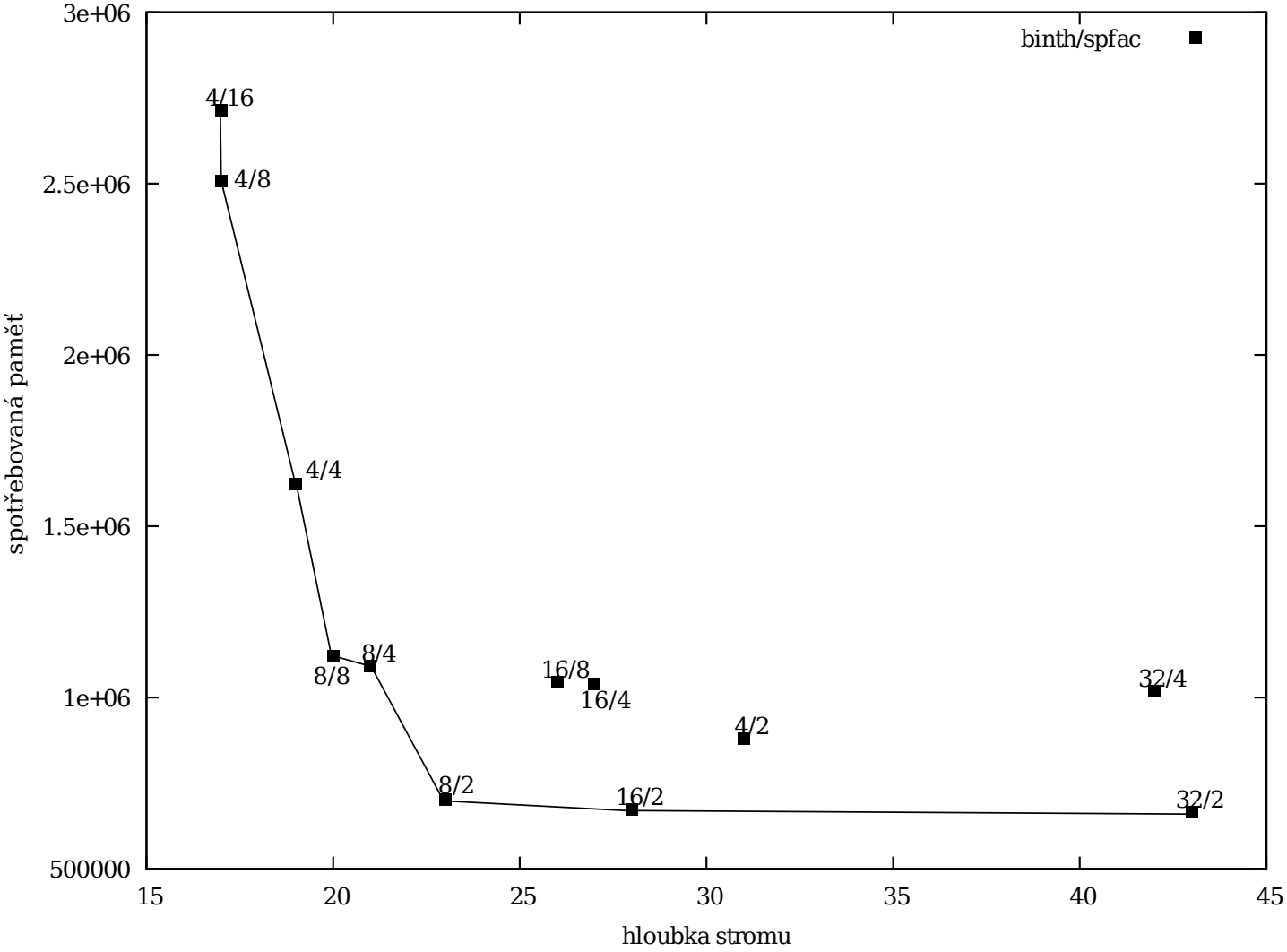
Tabulka 6.3: Tabulka Paretovy fronty pro spotřebovanou paměť (kbity) a hloubku stromu.

Množiny pravidel	MSCA	PHCA	PFCA	PCCA	DCFL	HiCuts
fw2_05_m05_250.rul	1 622, 91	553, 36	365, 80	227, 51	15, 69	193, 14
fw2_05_m05_500.rul	2 928, 36	874, 83	682, 13	601, 23	25, 12	367, 78
rules2.rul	162, 26	5 983, 96	5 310, 92	1 075, 99	14, 76	871, 24
rules3.rul	2 303, 81	21 362, 40	4 403, 81	3 347, 68	11, 08	2907, 63
rules4.rul	211, 68	865, 93	26, 75	260, 02	30, 31	20232, 19
rules6.rul	898, 77	1 306, 65	169, 30	814, 27	93, 67	1044, 05

Tabulka 6.4: Spotřeba paměti (kbity) algoritmů, u různých množin pravidel.



Obrázek 6.1: Graf porovnávající spotřebu paměti (v bitech) s a bez použití heuristiky znovupoužití uzlů. U osy spotřebované paměti je použito logaritimické měřítko.



Obrázek 6.2: Paretova fronta hloubky stromu a spotřebované paměti (v bitech), u algoritmu s použitou heuristikou.

Kapitola 7

Závěr

V této bakalářské práci byla popsána architektura počítačových sítí. Také byla vysvětlena problematika klasifikace paketů a její využití při filtrování provozu v počítačové síti.

Byly popsány vybrané klasifikační algoritmy z experimentální knihovny Netbench. U nich byly vysvětleny jejich základní principy a uvedeny výhody a nevýhody těchto algoritmů.

Dále byl detailně prostudován klasifikační algoritmus Hierarchical Intelligent Cuttings (HiCuts). Je vysvětlena základní konstrukce rozhodovacího stromu a jeho datová struktura. Zde byly uvedeny i heuristické funkce, používající se při tvorbě rozhodovacího stromu a zlepšení vlastnosti tohoto klasifikačního algoritmu.

Tento algoritmus byl posléze implementován tak, aby byl kompatibilní s experimentální knihovnou Netbench. Program umožňuje načíst soubor s klasifikačními pravidly, podle nich vytvořit rozhodovací strom. Následně přijímá pakety a na základě vytvořeného rozhodovacího stromu pakety klasifikuje. Program umožňuje tvorbu rozhodovacího stromu parametrizovat. V implementaci byly použity všechny zmíněné heuristické funkce.

Poté byly provedeny experimenty se zaměřením na paměťovou náročnost a hloubku rozhodovacího stromu. Byla porovnána paměťová náročnost algoritmu HiCuts s jinými klasifikačními algoritmy, pro různé množiny pravidel. Pomocí Paretovy fronty byly nalezeny parametry pro nejlepší vlastnosti rozhodovacího stromu.

Literatura

[1] David E. Taylor: *Survey and Taxonomy of Packet Classification Techniques.* Washington University in Saint Louis, 2004.

[2] David E. Taylor, Jonathan S. Turner: *Scalable Packet Classification using Distributed Crossproducting of Field Labels.* IEEE INFOCOM, 2005.

[3] Pankaj Gupta, Nick McKeown: *Algorithms for Packet Classification.* Computer Systems Laboratory, Stanford University.

[4] Pankaj Gupta, Nick McKeown: *Packet Classification using Hierarchical Intelligent Cuttings.* Computer Systems Laboratory, Stanford University, 2000.

[5] Petr Matoušek: *Architektura sítí, adresování, konfigurace TCP/IP.* FIT VUT Brno, 2011.

[6] Petr Matoušek: *Zajištění kvality služeb.* FIT VUT Brno, 2011.

[7] Sarang Dharmapurikar, Haoyu Song, Jonathan Turner, John Lockwood. : *Fast Packet Classification Using Bloom filters.* Washington University in St. Louis.

[8] Sartaj Sahni, Kun Suk Kim, Haibin Lu: *Data Structures For One-Dimensional Packet Class Most-Specific-Rule Matching.* Information Science and Engineering University of Florida.

[9] Viktor Puš: *Klasifikace Paketů S Využitím Technologie FPGA, Diplomová práce.* FIT VUT Brno, 2008.

[10] Viktor Puš, Jan Kořenek.: *Fast and Scalable Packet Classification Using Perfect Hash Functions.* Faculty of Information Technology Brno University of Technology.

[11] WWW stránky: Bloomův filtr. `http://en.wikipedia.org/wiki/Bloom_filter`.

[12] WWW stránky: Firewall. `http://cs.wikipedia.org/wiki/Firewall`.

[13] WWW stránky: Hardware Implementations of an Efficient Packet Classification Algorithm with an Incremental Update Capability. `www.hindawi.com/journals/ijrc/2011/648483/`.

[14] WWW stránky: ISO/OSI model. `http://en.wikipedia.org/wiki/OSI_model`.

[15] WWW stránky: Netbench. `http://merlin.fit.vutbr.cz/ant/netbench/index.html`.

[16] WWW stránky: Paretova fronta.
http://en.wikipedia.org/wiki/Pareto_efficiency.

[17] WWW stránky: Perfect Hash Crossproduct Algorithm.
http://merlin.fit.vutbr.cz/ant/technology/packet_classification.html.

[18] WWW stránky: PyLint. http://www.logilab.org/857.

Příloha A

Obsah CD

Na médiu je uložena technická zpráva ve formátu PDF, návod pro instalaci a spuštění programu, zdrojový kód programu, zdrojový tvar technické zprávy.