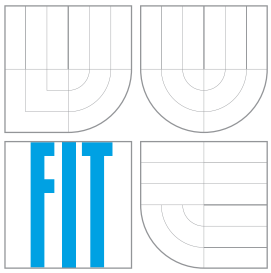
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

HARDWAROVÁ AKCELERACE APLIKACÍ PRO MONITOROVÁNÍ A BEZPEČNOST VYSOKORYCHLOSTNÍCH SÍTÍ

HARDWARE ACCELERATION OF NETWORK SECURITY AND MONITORING APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. LUKÁŠ KEKELY
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KOŘENEK, Ph.D.
SUPERVISOR

BRNO 2013

Zadání diplomové práce/15490/2012/xkekel00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2012/2013

Zadání diplomové práce

Řešitel: **Kekely Lukáš, Bc.**
Obor: Inteligentní systémy
Téma: **Hardwarová akcelerace aplikací pro monitorování a bezpečnost vysokorychlostních sítí**
Hardware Acceleration of Network Security and Monitoring Applications
Kategorie: Počítačové sítě

Pokyny:

1. Nastudujte problematiku monitorování a bezpečnosti počítačových sítí.
2. Dále se seznamte s technologií OpenFlow a perspektivní oblastí softwarově definovaného zpracování síťového provozu (SDN - Software Defined Networking).
3. Navrhněte systém pro softwarově řízenou hardwarovou akceleraci tak, aby bylo možné rychle vytvářet monitorovací a bezpečností aplikace pro vysokorychlostní sítě s propustností 100 Gb/s.
4. Vytvořte simulační model systému a prostřednictvím simulace analyzujte chování pro tři vybrané aplikace z oblasti monitorování a bezpečnosti počítačových sítí.
5. Na základě simulačního modelu proveďte srovnání tří vybraných aplikací s hardwarovou akcelerací a bez hardwarové akcelerace.
6. V závěru diskutujte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Kořenek Jan, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 17. září 2012
Datum odevzdání: 22. května 2013

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

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Diplomová práce se zaobírá návrhem systému pro softwarově řízenou hardwarovou akceleraci ve vysokorychlostních sítích. Hlavní úlohou je umožnění jednoduchého přístupu k akceleraci pro různé aplikace z oblasti monitorování a bezpečnosti síti. Vytvářený systém je navržen s ohledem na nasazení při rychlostech 100 Gb/s a poskytuje vysokorychlostní zpracování dat v FPGA na síťové kartě spolu s flexibilním softwarovým řízením. Kombinací rychlosti hardwaru a flexibility softwaru je umožněno jednoduché tvoření komplexních a výkonných síťových aplikací. Dosažitelné urychlení tří vybraných monitorovacích a bezpečnostních aplikací je v práci ukázáno využitím simulačního modelu navrženého systému.

Abstract

This master's thesis deals with the design of software controlled hardware acceleration system for high-speed networks. The main goal is to provide easy access to acceleration for various network security and monitoring applications. The proposed system is designed for 100 Gbps networks. It enables high-speed processing on an FPGA card together with flexible software control. The combination of hardware speed and software flexibility allows easy creation of complex high-performance network applications. Achievable performance improvement of three chosen monitoring and security applications is shown using simulation model of the designed system.

Klíčová slova

simulace, hardware, SDN, softwarově definované zpracování síťového provozu, monitorování, bezpečnost, vysokorychlostní sítě

Keywords

simulation, hardware, SDN, Software Defined Networking, monitoring, security, high-speed networks

Citace

Lukáš Kekely: Hardwarová akcelerace aplikací pro monitorování a bezpečnost vysokorychlostních sítí, diplomová práce, Brno, FIT VUT v Brně, 2013

Hardwarová akcelerace aplikací pro monitorování a bezpečnost vysokorychlostních sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jana Kořenka, Ph.D. Informace mi též poskytli lidé ze sdružení Cesnet, zejména Ing. Viktor Puš, Ph.D.

.....
Lukáš Kekely
22. květen 2013

Poděkování

Chtěl bych poděkovat hlavně vedoucímu práce Ing. Janovi Kořenkovi, Ph.D. za jeho odbornou pomoc a rady při psaní této práce. Chtěl bych též poděkovat lidem ze sdružení Cesnet za spolupráci a poskytnuté informace. Speciální poďekování patří i Ing. Viktorovi Pušovi, Ph.D. za jeho odbornou pomoc.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam algoritmov

Kapitola 1

Úvod

Jedným z hlavných znakov dnešnej doby je neustále sa zrýchľujúci rozvoj vo všetkých odvetviach ľudskej činnosti. Uvedený trend je samozrejme možné pozorovať aj v oblasti výpočtovej techniky, ktorá predstavuje jednu z najrýchlejšie sa rozvíjajúcich oblastí dnešnej doby. K najvýznamnejším oblastiam v rámci výpočtovej techniky patrí neodmysliteľne oblasť sieťovej komunikácie a do nej spadajúca celosvetová počítačová sieť (internet). Okrem lacnej a rýchlej komunikácie dnes internet slúži aj na poskytovanie širokého sortimentu rôznych ďalších služieb užívateľom. Od elektronického obchodovania, reklamy, cez hranie hier až po spoznávanie nových ľudí na veľmi populárnych sociálnych sieťach. Výrazný a stále rastúci podiel má taktiež prenos multimediálneho obsahu vysokej kvality akým je napríklad vysielanie rôznych internetových televízií.

Význam počítačových sietí, aj napriek ich terajšiemu obrovskému rozšíreniu, naďalej neúprosne rastie. Pribúda počet zariadení schopných sieťovej komunikácie, počet služieb dostupných cez internet, počet užívateľov a predlžuje sa aj čas, ktorý každodenne trávia on-line. Veď kto už dneska nemá vo vrecku mobilný telefón alebo iné prenosné zariadenie schopné pripojenia sa do siete? Okrem rastu veľkosti užívateľskej základne stúpa tiež množstvo samotných dát prenášaných medzi užívateľmi pri komunikácii. Uvedené trendy majú za následok exponenciálny nárast objemu dát tečúcich sieťou – za posledných 5 rokov vzrástol objem sieťovej komunikácie 12 násobne. Hlavným dôsledkom uvedeného nárastu je neustála potreba rýchlejšej a výkonnejšej infraštruktúry zabezpečujúcej prenos sieťových dát. V dnešnej dobe sa preto vo vysokorýchlostných sieťach prechádza od využitia technológií s rýchlosťami 1 a 10 Gb/s na 40 a 100 Gb/s technológie.

Súčasťou sieťovej infraštruktúry sú okrem základných funkčných zariadení zabezpečujúcich správne nasmerovania a prenos dát od pôvodcu k cieľu aj zariadenia na zabezpečenie sledovania bezpečnosti siete a monitorovanie jej aktuálneho stavu. Aby boli tieto zariadenia schopné poskytovať kvalitné informácie o sieti je potrebné, aby ich výkonnosť nezaostávala za výkonnosťou zvyšku infraštruktúry. Preto pri nutnosti neustáleho zrýchľovania sietí je potrebné rovnakým tempom zrýchľovať aj zariadenia (aplikácie) zabezpečujúce monitorovanie a bezpečnosť . Práve návrhom vhodnej platformy na podporu dostatočne rýchlych monitorovacích a bezpečnostných sieťových aplikácií pre vysokorýchlostné siete s rýchlosťami 100 Gb/s a viac sa zaoberá táto práca.

Bežnou odpoveďou na vyššie popísané zrýchľovanie sieťových zariadení je využitie hardvérovo urýchľovaných riešení. Softvérové riešenia všeobecne pracujú na univerzálnych výpočtových architektúrach. Oproti nim je hlavnou výhodou hardvérovo urýchľovaných technológií špecializácia a optimalizácia hardvérovej architektúry pre potreby riešenie konkrétnej úlohy. Špecializácia hardvéru prináša najmä možnosť využiť paralelné a zreťazené spraco-

vania dát pomocou výpočtových jednotiek optimalizovaných práve pre potreby konkrétnej úlohy. Výsledkom špecializácie je potom zefektívnenie využívania hardvérových zdrojov, ktoré vedie k zníženiu ceny a zvýšeniu výkonu výsledného riešenia.

Využitie hardvérového urýchľovania však okrem zvýšenia výkonnosti vyvoláva aj niekoľko zásadných problémov. Hlavný z nich je spojený s úzkou špecializáciou vytvorených hardvérových architektúr, ktorá nie vždy vedie k dostatočne flexibilním riešeniam pre potreby stále sa vyvíjajúcich a meniacich sietí. Ďalším veľkým problémom je náročnosť vytvorenia robustného hardvérového riešenia zložitejších problémov, ktorá môže byť výrazne väčšia ako vytvorenie obdobného softvérového riešenia. Pre uvedené problémy spojené s čistou hardvérovou akceleráciou sa stále častejšie presadzuje používanie vhodného spojenia softvérových a hardvérových prvkov v rámci jednotného riešenia. Týmto spojením môžu vznikať riešenia s dostatočnou výkonnosťou vďaka akcelerácií výkonnostne kritických častí, ktoré sú zároveň vďaka doplneniu softvérom flexibilnejšie a jednoduchšie realizovateľné. Rozšírenou oblasťou využívajúcou vhodné spojenie softvéru a hardvéru v sieťach je v dnešnej dobe oblasť Software Defined Networking (SDN). Hlavnou myšlienkou SDN je spojenie inteligentného a sofistikovaného softvérového riadenia s relatívne jednoduchými hardvérovými zariadeniami.

Myšlienka rozdelenia funkcionality aplikácií medzi jednoduchý výkonný hardvér a sofistikovaný softvér je potenciálne využiteľná aj v oblasti monitorovania a bezpečnosti vysokorýchlostných sietí. Riešenie úloh akými sú napríklad detekcia anomálií a útokov na sieti či extrakcia a analýza informácií z aplikačných dát paketov, je v dnešných vysokorýchlostných sieťach pomocou štandardných prístupov problematické. Keďže ide o relatívne zložité problémy, ich hardvérové riešenie je neúnosne náročné a nákladné z pohľadu využitých zdrojov aj potrebného vývoja. Na druhej strane softvérové riešenia sú jednoduchšie, ale majú problém s dosahovanou výkonnosťou, ktorá často nepostačuje ani pre potreby 10 Gb/s sietí nieto ešte pre 100 Gb/s. Avšak vytvorením vhodnej hardvérovej akceleračnej platformy by malo byť možné ich výkonnosť výrazne zvýšiť. Kľúčom k výkonnosti môže byť softvérovo riadené hardvérové predspracovanie paketov umožňujúce aplikačne kontrolovanú stratu informácie a distribúciu záťaže. Teda hardvérová platforma poskytujúca len prostriedky na rôzne predspracovanie a rozdelenie prijímaných sieťových dát, využitie ktorých by bolo možné nastavovať podľa aktuálnych potrieb konkrétnej riešenej úlohy. Softvérové aplikácie by tak mali možnosť výrazne redukovať objem nimi spracovávaných dát a dosiahnuť tak vyššie priepustnosti.

Cieľom tejto diplomovej práce je teda návrh systému pre softvérovo riadenú hardvérovú akceleráciu, ktorý umožňuje podporu jednoduchého a rýchleho vytvárania výkonných monitorovacích a bezpečnostných aplikácií pre vysokorýchlostné siete s priepustnosťou aspoň 100 Gb/s. Okrem samotného návrhu softvérovej a hardvérovej časti systému je vytvorený a popísaný aj simulačný model tohto systému. Dôraz je kladený na schopnosť čo najrealistickejšej reprezentácie správania sa systému s ohľadom na parametre jeho výkonnosti pri použití v reálnych sieťach. V simulácii je tiež pozorované zlepšenie výkonnosti zvolených monitorovacích a bezpečnostných aplikácií s využitím navrhnutého systému oproti prípadu bez jeho vyžitia. Nad rámec zadania je začaté vytváranie realizácie navrhnutého systému konkrétnou implementáciou niektorých jeho hardvérových jednotiek. Vytvorenie a nasadenie celkovej realizácie navrhnutého systému predstavuje budúce pokračovanie tejto diplomovej práce.

Diplomová práca je rozdelená na 8 kapitol. Kapitola 2 obsahuje popis základných princípov fungovania dnešných počítačových sietí, rozbor problematiky monitorovania a bezpečnosti sietí a nakoniec základný popis myšlienok technológií OpenFlow a Software Defined

Networking. V kapitole 3 je predstavený konceptuálny návrh fungovania výsledného systému akcelerácie a jednotlivých jeho častí. Kapitola 4 obsahuje výsledky meraní rôznych parametrov prenosu dát na vysokorýchlostných linkách v reálnej sieti. Cieľom uvedených meraní je ukázať použiteľnosť a teoretickú efektivitu navrhovaného konceptu systému v reálnych podmienkach, prípadne odhaliť jeho možné slabiny. V kapitole 5 je uvedený konečný implementačný návrh akceleračného systému. Kapitola 6 sa zaoberá popisom vytvorenej implementácie softvérovej časti systému rozšírenej o simulačný model hardvérovej časti. Obsahuje aj popis zvolených prvkov hardvérovej časti implementovaných nad rámec zadania práce. V kapitole 7 sú uvedené výsledky simulácií chovania sa navrhnutého systému pre tri zvolené aplikácie z oblasti monitoringu a bezpečnosti sietí. Nakoniec je v kapitole 8 uvedené celkové zhrnutie obsahu a dosiahnutých výsledkov práce.

Kapitola 2

Teoretický rozbor

V kapitole sú stručne zhrnuté teoretické vedomosti a používané postupy, tvoriace základ praktickej časti práce popísanej v nasledujúcich kapitolách. Začiatok rozboru tvorí stručný popis základného modelu a princípu fungovania dnešných počítačových sietí. Potom nasleduje podrobnejší popis vlastností vybraných bežne používaných sieťových protokolov. Hlavný dôraz je pritom kladený na informácie, ktoré sú v dátach protokolov obsiahnuté a môžu byť využité pre potreby monitorovania a bezpečnosti. Následne je uvedený popis základných princípov monitorovania sietí so zameraním na monitorovanie založené na tokoch a rozbor protokolov NetFlow a IPFIX. Nasledujúca sekcia popisuje základy bezpečnosti počítačových sietí z pohľadu detekcie anomálií a útokov. Kapitola následne pokračuje základným zhrnutím myšlienok z oblasti Software Defined Networking a bližším popisom protokolu OpenFlow. Ukončenie tvorí sekcia popisujúca existujúcu platformu realizujúcu hardvérovú akceleráciu sieťových aplikácií nazývanú Hanic [1].

2.1 Princíp počítačových sietí

Väčšina textu sekcie, ak nie je uvedené inak, vychádza z poznatkov uvedených v knihe [2] a z mojej bakalárskej práce [3]. Pri ľubovoľnom druhu komunikácie, teda aj sieťovej komunikácie, musia byť prítomné prvky (zariadenia) medzi, ktorými táto komunikácia prebieha. Minimálne ide o prvok, ktorý vystupuje ako pôvodca správy a o prvok, ktorému je správa určená, teda príjemca správy. V počítačových sieťach sa takéto zariadenia označujú ako koncové (ang. end devices) a patria medzi ne napríklad osobné počítače, IP telefóny alebo sieťové tlačiarne. S koncovými zariadeniami sa väčšina užívateľov bežne stretáva a sú im dobre známe.

Samotný pôvodca a príjemca komunikácie nie sú pre úspešnú komunikáciu dostatoční, okrem nich musí vždy existovať aj spôsob, akým si budú navzájom odovzdávať informácie (správy). V sieťach sa o odovzdávanie správ pri komunikácii starajú zariadenia označované ako sprostredkovateľské (ang. intermediary devices). Sprostredkovateľské zariadenia sa starajú hlavne o bezporuchový prenos správ sieťou, ich správne nasmerovanie a čo možno najrýchlejšie doručenie na miesto určenia. Medzi tento druh zariadení patrí napríklad rozbočovač, smerovač alebo sieťový most. Väčšina bežných užívateľov sa so sprostredkovateľskými zariadeniami často nestretáva a väčšinou o ich prítomnosti a funkčnosti v sieti ani nevie.

Ďalším dôležitým aspektom, ktorý podmieňuje schopnosť vzájomnej komunikácie je nevyhnutnosť mať isté prenosové médium na šírenie správ komunikácie. V počítačových sie-

ťach sa v dnešnej dobe bežne využívajú rôzne druhy prenosových médií. Patria sem napríklad medené či optické káble alebo bezdrôtové rádiové vysielanie (WiFi). Prenosové média teda tvoria spoje medzi jednotlivými komunikujúcimi sieťovými zariadeniami a komunikujúce zariadenia musia byť schopné zakódovať správy do vhodného formátu prenositeľného po danom médiu (napr. elektrický signál na medenom kábli).

Zabezpečením spomenutých troch aspektov komunikácie sú zariadenia schopné vzájomne si odovzdávať správy. Nie sú však schopné porozumieť nimi prenášaným informáciám. Preto je ďalším významným aspektom na dosiahnutie plnohodnotnej komunikácie existencia istého systém riadenia priebehu komunikácie. V dnešných počítačových sieťach je základom riadenia sieťovej komunikácie model TCP/IP. Model TCP/IP definuje základné komunikačné protokoly, ktoré musia sieťové zariadenia dodržiavať, aby boli schopné úspešne spolu komunikovať.

TCP/IP model vznikol už v 70. rokoch minulého storočia, vytvorila ho organizácia DARPA pôvodne pre armádu Spojených štátov [4]. Dnes je využívaný v drvivej väčšine sietí a je na ňom postavený aj základ dnešného internetu. Ide o protokolový model, ktorý obsahuje súbor implementácií základných služieb potrebných pre zabezpečenie úspešnej komunikácie a spojenie medzi komunikujúcimi zariadeniami v sieti. Ďalej definuje základné pravidlá pre formát správ, adresovanie zariadení a prenos dát.

Základným znakom TCP/IP modelu je rozdelenie zodpovednosti za jednotlivé operácie spojené s prenosom dát komunikácie sieťou do abstraktných vrstiev (ang. layers). Každá z vrstiev má teda definovanú svoju funkcionalitu a rozhranie, ktorým je spojená s ostatnými vrstvami. Výhodou uvedeného vrstvového prístupu je výrazné zjednodušenie vývoja a začleňovania nových protokolov, služieb alebo zariadení do existujúcej siete. Vďaka pevnému deleniu na vrstvy stačí vždy implementovať len špecifickú funkcionalitu tej vrstvy, na ktorej daná nová služba pracuje a ostatné vrstvy môžu byť zachované v nezmenenej podobe. Táto vlastnosť sa dnes prejavuje napríklad fungovaním TCP/IP sietí nad rôznymi prenosovými médiami a to vždy len so zmenenou realizáciou najnižšej vrstvy modelu.

Vrstvy v TCP/IP modely sú celkovo štyri. Číslujú sa od jednotky pre najnižšiu vrstvu, teda vrstvu najbližšie k hardvéru a prenosovému médiu. Každá z vrstiev pridáva k prenášaným dátam vlastné dodatočné informácie, takzvanú hlavičku (na začiatok) a prípadne aj pätu (na koniec). Tento proces sa nazýva zabaľovanie dát (ang. data encapsulation). Na strane odosielateľa sú vrstvy pri zabaľovaní dát prechádzané zhora dole. Na strane príjemcu pri rozbaľovaní dát sú vrstvy prechádzané opačne – zdola hore. Model TCP/IP obsahuje konkrétne nasledujúce štyri vrstvy:

Aplikačná vrstva je najvyššou (štvrtou) vrstvou TCP/IP protokolu. Jej úlohou je predovšetkým zabezpečovať vhodnú reprezentáciu a zobrazenie dát užívateľovi. Taktiež zabezpečuje kontrolu a definuje pravidlá priebehu dialógu medzi komunikujúcimi koncovými zariadeniami. Na tejto vrstve pracujú protokoly jednotlivých sieťových aplikácií. Tieto aplikácie väčšinou vnímajú nižšie vrstvy modelu len ako istú čiernu skrinku schopnú prenášať ich komunikáciu počítačovou sieťou medzi zariadeniami. Väčšina z aplikácií komunikuje na princípe klient-server a medzi najznámejšie protokoly patria napríklad HTTP, FTP, Telnet, SSH ...

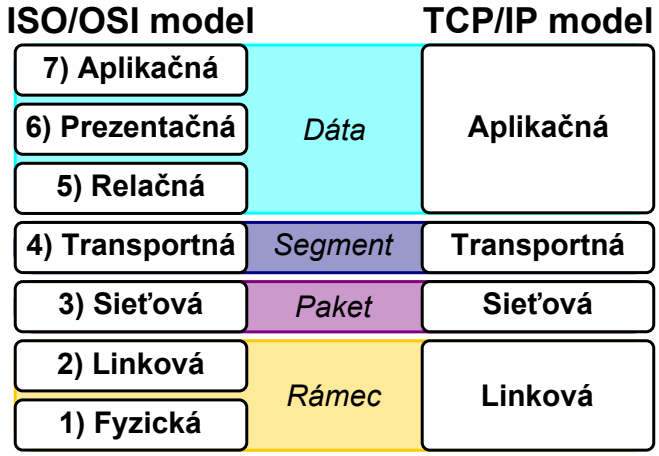
Transportná vrstva zabezpečuje spojenie medzi dvomi koncovými zariadeniami v sieti. Toto spojenie je nezávislé od použitých typov prenosových médií. Transportná vrstva tiež definuje mechanizmy kontroly zahltenia a toku správ, segmentáciu správ a kontrolu správnosti dát. Dôležitou je aj možnosť adresovania konkrétnych aplikácií bežiacich na jednom koncovom zariadení. Transportná vrstva sa preto často označuje ako,

vrstva spájajúca aplikácie. Najrozšírenejšími protokolmi pracujúcimi na transportnej vrstve sú protokoly TCP a UDP. TCP je protokol spájaný, ktorý zabezpečuje spoľahlivý prenos dát a zabezpečuje ich správnosť a integritu. Jeho nevýhodou však je relatívne vysoká réžia spojená s prenosom. Naproti tomu protokol UDP je nespájaný, má nižšiu réžiu, avšak je nespoľahlivý, teda neposkytuje žiadne mechanizmy kontrolujúce správnosť prenosu dát. Prenášané dátové entity (správy) sa na transportnej vrstve nazývajú segmenty (ang. segment).

Sieťová vrstva sa stará o zabezpečenie správneho doručovania segmentov medzi zariadeniami a hľadanie najlepšej cesty pri doručovaní segmentov medzi sieťami. Tento proces sa označuje smerovanie a nie je spoľahlivý. Základnú funkčnosť vrstvy zabezpečuje protokol IP starajúci sa o adresovanie a spojenie jednotlivých komunikujúcich zariadení. Funkčnosť IP protokolu je podporovaná ďalšími protokolmi, medzi ktoré patria napríklad ICMP (správy o chybách a výnimočných stavoch), IGMP (správa multicastových skupín) a tiež smerovacie protokoly (EIGRP, OSPF, RIP . . .). Prenášané dátové entity sú na sieťovej vrstve nazývané pakety (ang. packet).

Linková vrstva zabezpečuje ovládanie hardvérových zariadení a prenosových médií počítačových sietí. Popisuje postupy použité na kódovanie paketov na jednotlivých typoch prenosového média ako aj samotné pravidlá ich prenosu medzi zariadeniami pripojenými na rovnakej linke. Známe protokoly realizujúce linkovú vrstvu sú napríklad Ethernet a Token Ring. Prenášané dátové entity sú nazývané rámce (ang. frame).

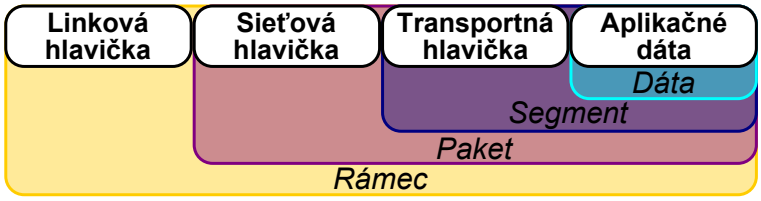
Okrem vyššie uvedeného modelu TCP/IP sa na popis fungovania počítačových sietí často používa aj model ISO/OSI, ktorý je, rovnako ako TCP/IP, vrstvovým modelom. Model ISO/OSI však nie je protokolovým modelom, ale referenčným. Jeho úlohou nie je špecifikácia implementácie a detailov jednotlivých vrstiev, ale skôr slúži ako pomoc na pochopenie funkcií a procesov využívaných v sieťovej komunikácii. ISO/OSI model pozostáva zo siedmych vrstiev, ktoré sú číslované rovnakým spôsobom ako pri TCP/IP.



Obr. 2.1: Vzájomné porovnanie vrstvových modelov TCP/IP a ISO/OSI

Približné mapovanie funkčnosti zabezpečovanej jednotlivými vrstvami oboch modelov spolu s názvami používanými na označenie dátových entít na jednotlivých vrstvách je zobrazené na obrázku 2.1. Najvyššia vrstva TCP/IP modelu spája funkčnosť najvyšších troch vrstiev ISO/OSI modelu. Vo väčšine dnešných sieťových aplikácií sa funkčnosť spomínaných

troch vrstiev prelína a nie je možné určiť pevné hranice medzi nimi. Transportné a sieťové vrstvy sú v oboch modeloch funkčne približne ekvivalentné. Funkčnosť najnižšej vrstvy TCP/IP modelu spája funkčnosť najnižších dvoch vrstiev ISO/OSI modelu.



Obr. 2.2: Zabalenie aplikačných dát v TCP/IP modele

Ako už bolo spomínané vyššie, pri komunikácii počítačovou sieťou sú prenášané dáta istým spôsobom zabaľované. Pri zabaľovaným dát na každej vrstve TCP/IP modelu vzniká vo výsledku rámec so štruktúrou zachytenou na obrázku 2.2. Na uvedenom obrázku sú znázornené len hlavičky pridávané protokolmi jednotlivých vrstiev modelu. Všeobecne je možné aby protokoly okrem hlavičiek pridávali aj päty, ale väčšina protokolov to nevyužíva. Hlavičky jednotlivých protokolov majú istým spôsobom pevne definovanú svoju štruktúru. Väčšinou sú rozdelené do niekoľkých políčok (ang. fields), kde každé políčko má daný svoj význam a dĺžku. Obsah políčok hlavičiek jednotlivých vrstiev teda predstavuje isté dodatočné a štruktúrované informácie o aplikačných dátach prenášaných v rámci.

2.2 Popis vybraných sieťových protokolov

Z pohľadu zamerania tejto práce sú spôsoby fungovania protokolov a informácie prenášané v jednotlivých hlavičkách rámca zaujímavé, pretože sa o ne často opierajú monitorovacie a bezpečnostné aplikáciách podrobnejšie popísaných v nasledujúcej sekcii. Je preto vhodné zamerať sa na podrobnejší popis vybraných bežne používaných protokolov a informácií nesených v ich hlavičkách. V rámci internetu najrozšírenejšími a v nasledujúcej časti textu popísanými protokolmi sú IP (sieťová vrstva) a TCP, UDP (transportná vrstva). Text s popisom zvolených protokolov uvedený v tejto skecii sa opiera o informácie z knihy [2] a zdroje odkazované priamo v texte.

Protokol sieťovej vrstvy IP (Internet Protocol) bol prvýkrát definovaný už v roku 1981 v RFC 791 [5]. Jeho hlavnou úlohou je zabezpečenie adresovania zariadení a vhodné smerovanie dát medzi nimi. Protokol IP teda implementuje základnú funkcionalitu definovanú sieťovou vrstvou TCP/IP modelu. Neobsahuje ale žiadne pokročilejšie mechanizmy spojené s prenosom dát ako je napríklad sledovanie toku alebo zaručenie spoľahlivého doručovania. Uvedenú funkcionalitu, ak je potrebná, musia zariadiť protokoly vyšších vrstiev fungujúce nad IP. Na druhej strane má však vďaka svojej jednoduchosti relatívne nízku réžiu prenosu segmentov. V dnešnej dobe je IP najrozšírenejšie používaným protokolom na doručovanie dát v počítačových sieťach.

Jedným z hlavných rysov IP je nespájaný typ komunikácie, teda pred začiatkom sieťovej komunikácie zariadení nie je medzi nimi potrebné dopredu vytvárať žiadne dedikované spojenie. Príjemca nie je dopredu upozorňovaný na príchod paketov a tie cestujú sieťou každý individuálne. Zároveň nie je dopredu jasné, či v sieti požadovaný príjemca vôbec existuje a či k nemu vedie nejaká cesta od odosielateľa. Preto môže počas cesty sieťou dôjsť ku strate niektorých IP paketov alebo k ich príchodu v inom poradí v akom boli odo-

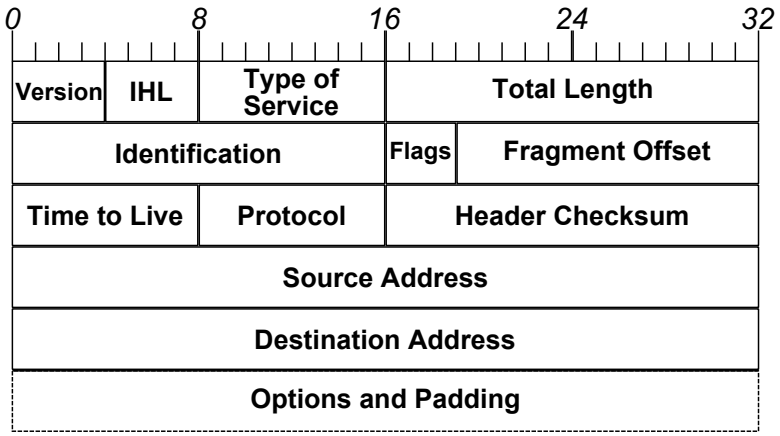
slané. Proces doručovania IP paketov sa dá prirovnať k princípu akým funguje doručovanie zásielok štandardnou poštou.

Ďalším dôležitým rysom IP protokolu je doručovanie s najlepším úsilím (ang. best effort). Ide o transport paketov medzi zariadeniami so snahou vytvoriť čo najmenšiu celkovú záťaž na sieť aj za cenu občasnej straty paketu. Každé zariadenie sa však podľa svojich aktuálnych možností snaží o čo najlepší posun každého prijatého paketu stále bližšie smerom k jeho cieľu. Povolenie občasných strát paketov je jedným z dôvodov nespoľahlivosti IP protokolu, nie všetky sieťové aplikácie však vyžadujú spoľahlivý prenos.

Posledným významným rysom IP protokolu je nezávislosť na prenosovom médiu. O vyriešenie závislosti na médiu sa z väčšej mieri starajú protokoly nižšej (linkovej) vrstvy. Z typu použitého prenosového média však vyplýva jedno obmedzenie aj pre protokol IP, konkrétne ide o obmedzenie vzťahujúce sa na maximálnu povolenú veľkosť prenášaných paketov. IP protokol na vyriešenie problému maximálnej veľkosti implementuje podporu rozdeľovania (fragmentovania) a následného znovu poskladanie paketov pri prenose sieťou.

V dnešnej dobe sa stretávame s dvomi rozšírenými a bežne používanými verziami protokolu IP. Jedná sa o verzie IPv4 a IPv6 [6]. Rozšírenejšou je dnes ešte stále verzia 4, ale podiel využitia verzie 6 neustále rastie. Hlavným dôvodom prechodu na verziu 6 je postupné vyčerpanie adresného priestoru poskytovaného IPv4 adresami. Dĺžka každej IPv4 adresy používanej na adresovanie jedného koncového zariadenia v sieti je 32 bitov, maximálny teoretický počet adresovateľných zariadení je preto 2^{32}, čo je viac ako 4 miliardy adries. Tento počet však v dnešnej dobe už nestačí. Naproti tomu v IPv6 je dĺžka adresy 128 bitov a poskytuje teoretický počet rôznych adries zariadení rádovo 10^{38}.

Vo verzii 4 protokolu IP je 32 bitová adresa pri zápise rozdelená na štyri 8 bitové čísla (ang. octets) navzájom oddelené bodkami, ktoré sa zapisujú v dekadickom tvare. Príkladom zápisu IPv4 adresy je napríklad adresa 192.168.1.2. Vo verzii 6 je 128 bitová IPv6 adresa [7] pri zápise rozdelená na osem 16 bitových skupín navzájom oddelených dvojbodkami, ktoré sa zapisujú v hexadecimálnom tvare. Príkladom zápisu IPv6 adresy je napríklad adresa 2001:0DB8:0000:0000:0000:0000:1428:0001. IPv6 adresy je možné zapísať aj v skrátenej forme, kedy je možné jednu ľubovoľnú neprerušenú postupnosť nulových skupín vynechať. Vyššie spomenutú ilustračnú IPv6 adresu je teda možné zapísať v skrátenom tvare ako 2001:0DB8::1428:0001. Takisto počiatočné nuly v každej skupine je možné vynechávať, zápis ilustračnej adresy je preto ešte možné skrátiť na tvar 2001:DB8::1428:1.



Obr. 2.3: Schéma štruktúry IPv4 hlavičky

Z pohľadu štruktúry a informácií obsiahnutých v sieťovej hlavičke sa verzia 4 a 6 protokolu IP navzájom líšia. Niektoré významné informácie sú však obsiahnuté v oboch verziách hlavičiek aj keď v rôznej forme (napr. rôzny tvar IPv4 a IPv6 adries) alebo rôzne označené. Hlavičky oboch verzií sú delené na políčka istej dĺžky a významu, kde jednotlivé políčka obsahujú hodnoty zapísané v binárnej podobe.

Schému štruktúry IPv4 hlavičky je možné vidieť na obrázku 2.3 [5]. Číselná stupnica v hornej časti obrázku slúži na odčítanie dĺžok jednotlivých políčok hlavičky a je uvedená v bitoch. Jednotlivé políčka hlavičky sú reprezentované obdĺžnikmi s anglickým názvom položky vpísanom vnútri. Pre potreby monitorovacích a bezpečnostných aplikácií sú zo zobrazenej IPv4 hlavičky významné najmä nasledujúce políčka:

Version: políčko indikujúce formát hlavičky určením čísla verzie IP protokolu. Pre IPv4 by mala byť hodnota nastavená na 4.

IHL (Internet Header Length): určuje dĺžku IP hlavičky v počte 32 b (4 B) slov. Minimálna hodnota pre platnú IPv4 hlavičku je 5 (20 B). Nastavená dĺžka hlavičky sa prejavuje zmenou dĺžky políčka Options and Padding.

Type of Service (ToS): políčko obsahujúce hodnotu definujúcu prioritu paketu. Hodnota políčka ToS umožňuje využitie mechanizmu zaistenia kvality prenosu (QoS), kedy je možné uprednostniť spracovanie niektorých paketov.

Total Length: určuje dĺžku IP paketu (hlavička aj dáta) v počte bajtov. Maximálna dĺžka IP paketu môže byť až 65 535 B, odporúča sa však používať výrazne kratšie pakety.

Flags: pozostáva z jednobitových príznakov ovplyvňujúcich fragmentovanie paketu. Obsahuje príznak na zakázanie/povolenie fragmentovania a príznak označujúci posledný fragment.

Protocol: definuje typ dát nesených v IP pakete. Obvykle sa jedná o identifikátor niektorého protokolu vyššej vrstvy (napr. 6 pre TCP alebo 17 pre UDP).

Source Address: políčko obsahujúce IPv4 adresu (identifikáciu) príjemcu paketu.

Destination Address: políčko obsahujúce IPv4 adresu (identifikáciu) pôvodcu paketu.

Schému štruktúry IPv6 hlavičky je možné vidieť na obrázku 2.4 [6]. Číselná stupnica v hornej časti obrázku slúži na odčítanie dĺžok jednotlivých políčok hlavičky a je uvedená v bitoch. Políčka hlavičky sú reprezentované obdĺžnikmi s anglickým názvom políčka vpísanom vnútri. Pre potreby monitorovacích a bezpečnostných aplikácií sú zo zobrazenej IPv6 hlavičky významné najmä nasledujúce políčka:

Version: políčko určujúce verziu IP protokolu. Pre IPv6 by mala byť hodnota nastavená na 6.

Traffic Class: políčko určujúce triedu alebo prioritu paketu, má podobný význam ako políčko ToS v IPv4 hlavičke.

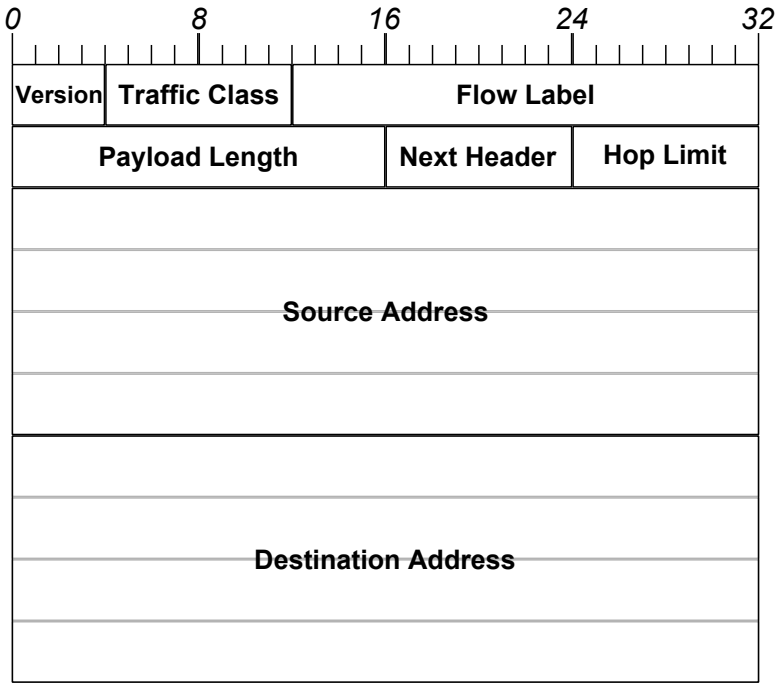
Payload Length: určuje dĺžku dát v IP pakete (bez hlavičky) v počte bajtov. Dáta paketu začínajú hneď za IPv6 hlavičkou.

Next Header: obsahuje identifikátor typu hlavičky nasledujúcej hneď za IP hlavičkou. Má podobnú funkciu ako políčko Protocol v IPv4 hlavičke a tiež používa rovnaké hodnoty identifikátorov protokolov.

Source Address: políčko obsahujúce IPv6 adresu (identifikáciu) príjemcu paketu.

Destination Address: políčko obsahujúce IPv6 adresu (identifikáciu) pôvodcu paketu.

Okrem základnej IPv6 hlavičky popísanej štruktúry definuje protokol IPv6 aj možnosť požitia rozširujúcich hlavičiek (ang. Extension Headers).



Obr. 2.4: Schéma štruktúry IPv6 hlavičky

Rozširujúce IPv6 hlavičky môžu byť umiestnené priamo za základnú IPv6 hlavičku (pred hlavičku protokolu vyššej vrstvy) a nesú voliteľné rozširujúce informácie (podobne ako pole Options v IPv4). Každý IPv6 paket môže obsahovať nula, jednu alebo viac rozširujúcich hlavičiek radených za sebou, pričom poradie nie je pevne špecifikované. Každá rozširujúca hlavička začína políčkom určujúcim dĺžku tejto hlavičky a políčkom definujúcim typ nasledujúcej hlavičky (Next Header). V špecifikácii protokolu IPv6 sú zadefinované aj základné typy zavedených rozširujúcich hlavičiek a ich identifikátory. Okrem informácií nesených priamo v rozširujúcich hlavičkách môže istú informáčnú hodnotu predstavovať aj samotná prítomnosť daných typov týchto hlavičiek v pakete.

Medzi najpoužívanejšie protokoly transportnej vrstvy v počítačových sieťach patria protokoly TCP (Transmission Control Protocol, RFC 793 [8]) a UDP (User Datagram Protocol, RFC 768 [9]). Oba uvedené protokoly poskytujú základnú funkcionalitu požadovanú od transportnej vrstvy, teda sú schopné medzi sebou rozlišovať a adresovať jednotlivé komunikujúce aplikácie bežiace súčasne na jednom koncovom zariadení. Líšia sa však v podpore kontroly priebehu tejto komunikácie.

Protokol UDP je veľmi jednoduchým protokolom. Jeho jednoduchosť mu umožňuje dosiahnuť doručovanie dát s nízkou réžiou – krátka hlavička a jednoduchá funkcionalita. Využíva doručovanie s najlepším úsilím poskytované už protokolom sieťovej vrstvy bez ďalších pokročilejších mechanizmov na zaistenie spoľahlivého či bezproblémového prenosu. Poskytovanie nespoľahlivého prenosu mu umožňuje fungovať v nespájanom a bezstavovom režime,

kedy nie je potrebné žiadne naviazanie, ukončenie ani sledovanie stavu spojenia medzi komunikujúcimi prvkami. Neprítomnosť sledovania stavu spojenia vedie k nemožnosti podpory riadenia toku a prevencie zahltenia siete pri použití protokolu UDP.

Naproti protokolu UDP je protokol TCP podstatne zložitejší. Jeho zložitosť sa prejavuje relatívne vysokou réžiou doručovania dát – dlhšia hlavička a zložitejšia funkcionalita. Vysoká réžia umožňuje protokolu TCP na druhej strane podporovať rozsiahlejšiu paletu pokročilejších vlastností. Hlavnou vlastnosťou TCP je poskytovanie mechanizmu spoľahlivého doručovania dát nad všeobecne nespoľahlivou sieťovou vrstvou. Mechanizmus spoľahlivého doručovanie dát v sebe spája dve základné vlastnosti. Prvou je zaistenie doručenia všetkých odosielaných dát príjemcovi pomocou mechanizmov opätovného preposielania stratených častí dát odosielateľom a pozitívneho potvrdzovania prijatých častí príjemcom. Druhou vlastnosťou je doručenie dát v správnom poradí, kedy v prípade príchodu segmentov k príjemcovi v zlom poradí sa TCP sám postará o ich správne zoradenie.

Poskytovanie pokročilejšej funkcionality spojenej so spoľahlivým doručovaním vyžaduje, aby protokol TCP pracoval v spájanom a stavovom režime. Spájaný režim vyžaduje, aby samotnej komunikácii predchádzal proces ustanovenia spojenia medzi odosielateľom a príjemcom. Podobne po skončení prenosu dát je potrebné vytvorené spojenie ukončiť. Na ustanovenie a ukončenie spojenia slúžia v TCP špeciálne pakety označené príznakmi v hlavičke, tieto príznaky sú bližšie popísané v nasledujúcom texte pri rozbore hlavičky. Vďaka sledovaniu stavu spojenia je TCP schopný poskytnúť mechanizmus riadenia toku dát, ktorého hlavnou úlohou je pomocou vhodného dynamického obmedzovania počtu po sebe odosielaných segmentov (veľkosti okna) zamedziť zahltenie siete. Prevencia zahltenia vedie k zníženiu pravdepodobnosti zahadzovania paketov v sieti a znižuje tak potrebný počet znovu posielaných segmentov.

Protokoly UDP a TCP využívajú na adresovanie a rozlíšenie jednotlivých komunikácii (komunikujúcich aplikácií) čísla portov. Číslo portu je 16 bitové číslo bez znamienka, teda v rozsahu 0 až 65 535. Každá komunikujúca aplikácia na zariadení má priradené unikátne číslo portu, ktorým sú označované všetky segmenty nesúce dáta patriace tejto aplikácii. Keďže sieťová komunikácia prebieha medzi dvomi aplikáciami, je potrebné, aby každý segment obsahoval 2 čísla portov. Jedno identifikujúce komunikujúcu aplikáciu na strane odosielateľa a druhé na strane príjemcu. Z pohľadu prideľovania aplikáciám je možné čísla portov rozdeliť do nasledujúcich skupín:

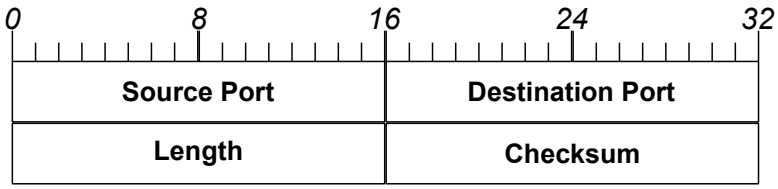
Známe porty (0 až 1023): (ang. Well Known Ports) sú čísla rezervované pre bežne používané služby a aplikácie. Prideľované sú organizáciou IANA a používajú sa na strane serverov poskytujúcich tieto služby. Využitím známych čísel portov je možné, aby sa klientske aplikácie vyžadujúce konkrétnu službu obracali na server s dopredu známym číslom portu. Inak povedané čísla portov tejto kategórie predstavujú implicitné porty používané na serveroch poskytujúcich konkrétnu aplikáciu. Príkladom môže byť štandardný port 80 používaný HTTP servermi alebo port 22 pre SSH server.

Registrované porty (1024 až 49 151): patria užívateľským serverovým procesom a aplikáciám, ktoré nie sú natoľko rozšírené (štandardné), aby im bol pridelený známy port. Okrem použitia pre serverové aplikácie je možné dynamicky prideľovať porty tohto rozsahu komunikujúcim klientskym aplikáciám.

Dynamické a privátne porty (49 152 až 65 535): tiež označované ako efemérne (krátkeho trvania), sa štandardne používajú na dynamické prideľovanie klientskym aplikáciám, ktoré zahajujú komunikáciu. Využitie týchto portov na strane serverových

aplikácií je veľmi zriedkavé, využívajú ich len niektoré programy na zdieľanie súborov.

Z pohľadu štruktúry a informácií obsiahnutých v transportnej hlavičke sa protokoly UDP a TCP navzájom podstatne líšia. Z významnejších informácií majú spoločné len čísla portov, ktoré sú v oboch protokoloch prítomné v rovnakom formáte aj na rovnakej pozícii v hlavičke. Hlavičky oboch protokolov sú podobne ako v protokole IP delené na políčka istej dĺžky a významu, kde jednotlivé políčka obsahujú hodnoty zapísané v binárnej podobe.



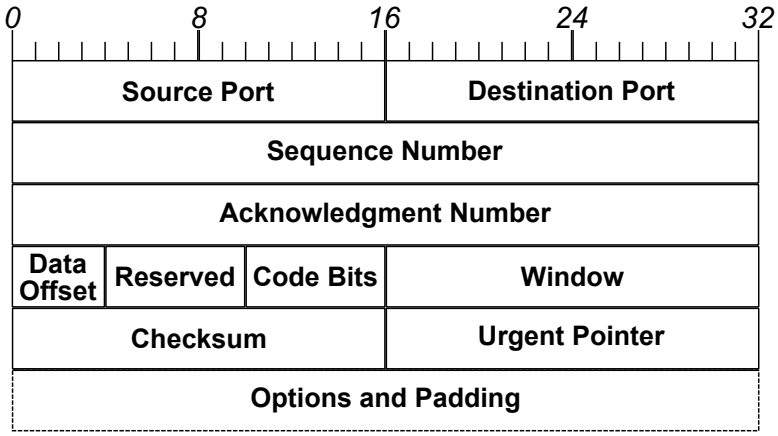
Obr. 2.5: Schéma štruktúry UDP hlavičky

Schému štruktúry UDP hlavičky je možné vidieť na obrázku 2.5 [9]. Číselná stupnica v hornej časti obrázku slúži na odčítanie dĺžok jednotlivých políčok hlavičky a je uvedená v bitoch. Políčka hlavičky sú reprezentované obdĺžnikmi s anglickým názvom políčka vpísanom vnútri. Pre potreby monitorovacích a bezpečnostných aplikácií sú zo zobrazenej UDP hlavičky významné najmä nasledujúce políčka:

Source Port: obsahuje číslo portu patriace aplikácii na lokálnom zariadení, ktorá je pôvodcom dát v segmente a predstavuje aj číslo portu, na ktorom je prípadne očakávaná odpoveď na tieto dáta. Hodnotu políčka nie je podľa štandardu nutné vždy vyplniť a môže tak ostať nastavené na nulu.

Destination Port: obsahuje číslo portu patriace aplikácii na vzdialenom zariadení, ktorej sú dáta v segmente určené.

Length: určuje dĺžku UDP segmentu (hlavička aj dáta) v počte bajtov. Minimálna platná hodnota (8) je daná dĺžkou samotnej UDP hlavičky.



Obr. 2.6: Schéma štruktúry TCP hlavičky

Schému štruktúry TCP hlavičky je možné vidieť na obrázku 2.6 [9]. Číselná stupnica v hornej časti obrázku slúži na odčítanie dĺžok jednotlivých políčok hlavičky a je uvedená

v bitoch. Políčka hlavičky sú reprezentované obdĺžnikmi s anglickým názvom políčka vpísanom vnútri. Význam jednotlivých políčok zobrazenej TCP hlavičky je nasledujúci:

Source Port: obsahuje číslo portu patriace aplikácii na lokálnom zariadení, ktorá je pôvodcom dát v segmente a predstavuje aj číslo portu, na ktorom je prípadne očakávaná odpoveď na tieto dáta. Hodnotu políčka nie je podľa štandardu nutné vždy vyplniť a môže tak ostať nastavené na nulu.

Destination Port: obsahuje číslo portu patriace aplikácii na vzdialenom zariadení, ktorej sú dáta v segmente určené.

Data Offset: určuje dĺžku TCP hlavičky v počte 32 b (4 B) slov. Minimálna hodnota pre platnú TCP hlavičku je 5 (20 B). Nastavená dĺžka hlavičky sa prejavuje zmenou dĺžky políčka Options and Padding.

Code Bits: kontrolné bity (príznaky) určujúce špeciálny význam segmentu. V poradí zľava doprava ide o nasledujúce bity:

URG - udáva platnosť políčka Urgent Pointer, označujú sa tak urgentné dáta nesené v segmentoch

ACK - udáva platnosť políčka Acknowledgment Number, označuje segmenty odosielané príjemcom dát ako potvrdenie správneho príjmu

PSH - vynucuje predanie dát aplikácii na strane príjemcu okamžite po prijatí segmentu (obchádza vyrovnávaciu pamäť), využíva sa najmä v aplikáciách vyžadujúcich komunikáciu v reálnom čase

RST - požiadavka na reštartovanie spojenia

SYN - žiadosť odosielateľa o synchronizáciu sekvenčných čísel, využíva sa najmä pri ustanovovaní nového spojenia

FIN - odosielateľ už nebude posielať ďalšie dáta, využíva sa pri ukončovaní existujúceho spojenia

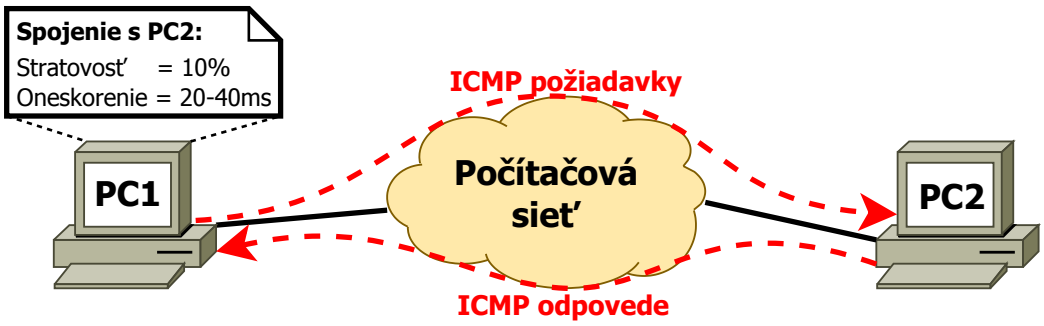
2.3 Monitorovanie sietí

Rýchly rozvoj vyžívania počítačových sietí má za následok neustály rast komplexnosti a zložitosti sieťovej infraštruktúry. Rôznorodosť využitia sietí zase vedie k existencii veľkého počtu paralelne fungujúcich služieb a protokolov rôznych vlastností a správania sa. Na celkovej zložitosti siete sa veľkou mierou podieľa aj dynamicky sa meniace usporiadanie siete, kedy sa napríklad mobilné koncové zariadenia môžu pripájať na rôznych miestach siete. Aby bolo možné takto komplikovaný systém rozumne spravovať a udržovať, je potrebné aby mali správcovia možnosť istým vhodným spôsobom nahliadnuť do aktuálnych dejov prebiehajúcich v spravovanej sietí. Dôležité je mať prehľad nielen o správaní sa a vyťažení samotných prvkov sieťovej infraštruktúry, ale aj o vlastnostiach dát tečúcich sieťou. Poskytovaním popísaného náhľadu do správania sa sietí sa zaoberá oblasť označovaná monitorovanie sietí (ang. network monitoring) prípadne meranie sietí (ang. network measurement) a na ňu nadväzujúca oblasť analýzy sieťovej komunikácie (ang. network traffic analysis). Popis základných princípov týchto oblastí uvedený v nasledujúcom texte vychádza zo zdrojov [10, 11, 12, 13].

Monitorovanie a analýza sietí je vo všeobecnosti proces neustáleho periodického zisťovania stavu prvkov siete a zbierania informácií o práve prebiehajúcich komunikáciách s cieľom čo najpresnejšie určiť čo sa v sieti práve odohráva. Monitorovaním získané informácie môžu byť následne využité správcom na zabezpečenie bezproblémovejšieho a efektívnejšieho fungovania siete. Informácie odvoditeľné z monitorovaním získaných dát je v zásade možné rozdeliť do dvoch základných skupín. Prvou skupinou sú informácie získavané v reálnom čase, teda napríklad parametre aktuálneho vyťaženia prvkov siete, vlastnosti aktuálne prebiehajúcich komunikácií a podobne. Na ich základe je možné rozpoznať a následne vhodne reagovať na rôzne anomálie v správaní sa zariadení a komunikácií na sieti akými sú napríklad zlyhania zariadení, či niektoré typy cielených sieťových útokov. Druhou skupinou informácií sú dlhodobejšie štatistiky získavané ukladaním a vhodným agregovaním periodicky sledovaných parametrov. Tieto informácie poskytujú náhľad na dlhodobé trendy vývoja používania siete a môžu tvoriť základ pre vhodné plánovanie jej budúceho rozvoja. Z pohľadu zamerania práce má väčší význam problematika monitorovania parametrov sieťovej komunikácie ako získavanie stavu funkčných prvkov siete, preto je nasledujúci text zameraný viac na uvedenú časť oblasti monitorovania.

Monitorovanie komunikácií prebiehajúcich v počítačovej sieti je možné realizovať rôznymi technikami. Základné rozdelenie techník monitorovania komunikácií do dvoch skupín je možné na základe toho, ktoré zariadenia realizujú získavanie informácií. Prvou skupinou v uvedenom delení sú techniky využívajúce na monitorovanie priamo prvky tvoriace funkčnú infraštruktúru siete. Výhodou tohto prístupu je nepotrebnosť zapojenia dodatočných monitorovacích zariadení do siete. Nevýhodami sú najmä nižšia robustnosť a flexibilita monitorovania plynúca z možnosti sledovať len výrobcom zariadenia podporovanú množinu parametrov, a tiež isté zníženie výkonnosti siete pridaním dodatočnej záťaže súvisiacej s monitorovaním na zariadenia funkčnej infraštruktúry. Druhou skupinou v uvedenom delení sú techniky monitorujúce sieť pomocou na to špeciálne vyhradených monitorovacích zariadení. Výhodami tohto prístupu sú väčšia flexibilita a robustnosť monitorovanie a minimalizovanie vplyvu monitorovacieho procesu na funkčné prvky infraštruktúry. Nevýhodou je potreba zapojenia dodatočného hardvéru do siete.

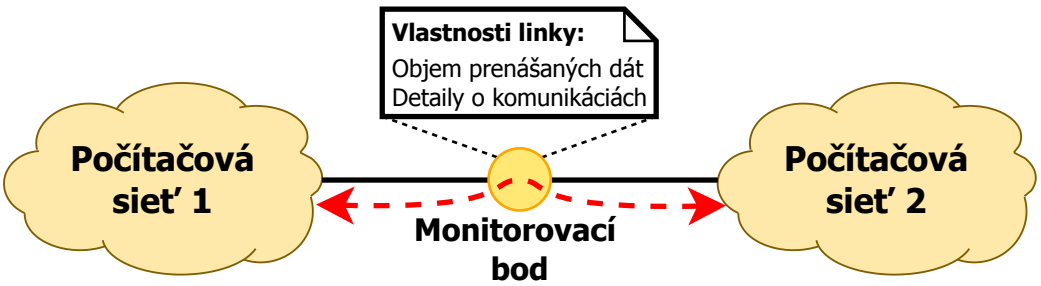
Druhým možným delením techník zisťovania parametrov sietí je delenie na základe aktívnosti monitorovacieho prístupu. Aktívnosť v tomto prípade označuje potrebu monitorovacej techniky odosielať do siete vlastné pakety. Z pohľadu uvedeného delenia môžu byť techniky buď aktívne alebo pasívne. Obe skupiny techník sú schopné použitím rôznych prostriedkov získať rôzne typy informácií o monitorovanej sieti. Zároveň je možné použitie oboch typov kombinovať na dosiahnutie robustnejšieho monitorovacieho systému.



Obr. 2.7: Príklad informácie získanej programom ping

Aktívne monitorovacie techniky využívajú mechanizmy, ktoré za účelom získavania informácií o sieti do nej vkladajú umelo generované komunikácie a sledujú reakciu siete na tieto komunikácie. Cieľom aktívnych techník je hlavne zistenie informácií o aktuálnej stavbe monitorovanej sieťovej infraštruktúry, oneskorení a priepustnosti prenosu dát, aktuálne používanom smerovaní paketov alebo stratovosti prenosu paketov. Na druhej strane však neposkytujú žiadnu informáciu o aktuálne prebiehajúcich komunikáciách v sieti a nimi generovaná komunikácia môže navyše interferovať s reálnou komunikáciou a ovplyvňovať tak celkové správanie sa meranej siete. Medzi najbežnejšie a najznámejšie nástroje na aktívne meranie parametrov siete patria programy ping a traceroute. Oba uvedené programy vysielajú do siete ICMP pakety a parametre siete určujú vyhodnotením prijatej odpovede na ne. Na obrázku 2.7 je zobrazený príklad použitia programu ping na zistenie kvality spojenia medzi zobrazenými počítačmi (z PC1 na PC2) cez sieť neznámej štruktúry (ilustrovaná mrakom). Výstupom programu sú zistené informácie o oneskorení a spoľahlivosti prenosu paketov ilustrované v obdĺžniku nad PC1.

Pasívne monitorovacie techniky na rozdiel od aktívnych nepotrebujú do siete vkladať žiadnu vlastnú komunikáciu a informácie získavajú len na základe v sieti už existujúcich komunikácií. Cieľom pasívnych techník je získanie informácií o aktuálne prebiehajúcich komunikáciách v sieti napríklad z pohľadu objemu prenášaných dát, parametrov komunikácií a ich paketov, zastúpenia jednotlivých protokolov alebo komunikujúcich užívateľov (koncových zariadení). Veľkou výhodou pasívneho prístupu je, že nemá priamy vplyv na správanie sa monitorovanej siete. Nevýhodou je však relatívne veľký objem dát, ktoré je potrebné spracovať a vhodne reprezentovať. Najjednoduchšími nástrojmi na pasívne monitorovanie siete sú rôzne programy na zachytávanie a ukladanie paketov ako napríklad wireshark alebo tcpdump, v sieťových prvkoch rôznych výrobcov zase býva často implementovaná podpora databáze MIB a protokolu SNMP. Na obrázku 2.8 je zobrazený príklad nasadenia pasívneho monitorovania v sieti. Monitorovací bod na obrázku sleduje pakety na linke spájajúcej 2 rôzne siete a spracúva zaujímavé informácie o komunikáciách prebiehajúcich medzi týmito sieťami.



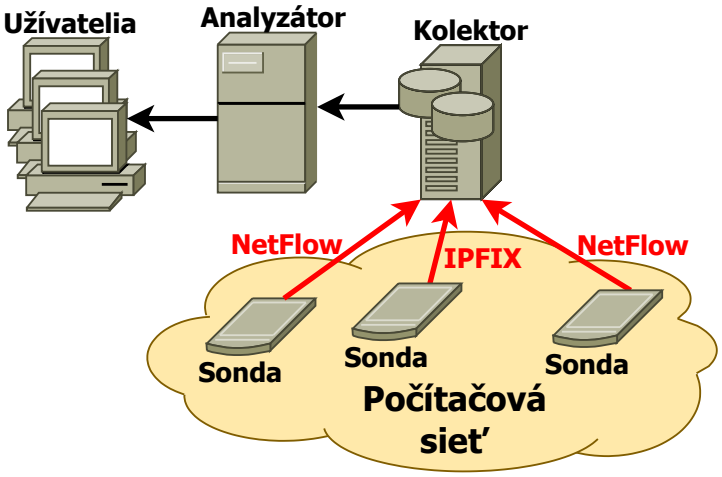
Obr. 2.8: Príklad nasadenia pasívneho monitorovania v sieti

Už vyššie je načrtnutý problém s veľkým objemom spracovávaných dát pri pasívnom monitorovaní. V dnešných sieťach je bežné, že jednou linkou tečú tisíce paketov s celkovým objemom v ráde jednotiek až desiatok gigabitov každú sekundu. Sledovanie a ukladanie celých paketov či len zaujímavých informácií o každom z nich je preto kapacitne príliš náročné a v praxi nedosiahnuteľné. Výrazné zníženie objemu nameraných dát je možné dosiahnuť počítaním a ukladaním len štatistických prípadne agregovaných informácií o zhlukoch paketov na linke namiesto ukladania informácií o jednotlivých paketoch. Pri použití štatistických informácií klesá informačná hodnota nameraných dát a niektoré zaujímavé vlastnosti tak môžu ostať skryté. Na druhej strane použitím vhodného agregovania informácií o paketoch

je možné docieliť zníženie objemu nameraných dát pri relatívne malej strate zaujímavých informácií. Ako vhodný spôsob agregovania sa veľmi často používa združovanie paketov do sieťových tokov a následné ukladanie len informácií o týchto tokoch.

Sieťový tok [13] je definovaný ako postupnosť paketov majúcich isté spoločné vlastnosti a prechádzajúcich pozorovaným bodom siete za určitý časový interval. Spoločné vlastnosti sú pritom odvodzované najčastejšie len na základe obsahu hlavičiek sieťových protokolov v paketoch. Využitie hlavičiek protokolov umožňuje nezávislosť definície toku na dátach prenášaných paketmi, a tým napríklad invariantnosť voči šifrovaniu obsahu. Konkrétne hodnoty zvolených spoločných vlastností paketov jedného toku identifikujú tento tok a zároveň ho odlišujú od ostatných. Pri monitorovaní založenom na tokoch sú ku každému toku okrem jeho identifikácie ukladané aj informácie agregované z jemu patriacich paketov. V tomto procese je každému pozorovanému paketu najprv na základe jeho vlastností správne priradený práve jeden konkrétny tok. Následne sú zaujímavé parametre tohto paketu započítané do agregovanej informácie patriacej priradenému toku. Na vytvorení informácie o istom toku sa tak istou časťou podieľajú všetky pozorované pakety rovnakých vlastností za určitý čas. Medzi základné ukladané informácie o tokoch patrí celková veľkosť toku (v paketoch a bajtoch) a časové údaje o trvaní toku (začiatočný a koncový čas). V poslednej dobe je rastúca snaha doplniť základné sledované vlastnosti tokov o ďalšie zaujímavé informácie a spresniť tak monitorovaním získavané dáta. Medzi zaujímavé položky sú často zaradzované informácie získateľné z hlavičiek aplikačných protokolov alebo precíznejšou analýzou paketov (napr. rozbaľovanie IP tunelov).

V dnešných počítačových sieťach založených na TCP/IP modeli sa na identifikáciu tokov najčastejšie používa pätica hodnôt z hlavičky sieťovej (IPv4 alebo IPv6) a transportnej (UDP alebo TCP) vrstvy. Konkrétne ide o zdrojovú a cieľovú IP adresu, číslo zdrojového a cieľového portu a identifikátor protokolu transportnej vrstvy (položka Protocol alebo Next Header IP hlavičky). Táto pätica býva občas doplnená ešte o položku ToS z IPv4 hlavičky alebo o identifikátor rozhrania monitorovacieho zariadenia. Každý tok tak reprezentuje jeden smer komunikácie istého typu, patriacej istej službe a prebiehajúcej medzi dvomi konkrétnymi koncových zariadeniami.



Obr. 2.9: Ilustrácia architektúry systému monitorovania tokov

Na realizovanie pasívneho monitorovania siete založeného na sledovaní sieťových tokov sa bežné používa monitorovacia architektúra ilustrovaná na obrázku 2.9. Zobrazená monitorovacia architektúra je zložená zo 4 základných prvkov:

Sonda sa stará o sledovanie a zachytávanie paketov na monitorovacom bode v sieti. Taktiež zabezpečuje zaraďovanie paketov do tokov, udržovanie záznamov s informáciami o tokoch a export týchto informácií na kolektor. Realizovaná býva štandardne formou špecializovaného zariadenia alebo komoditného počítača so špecializovanou vysokorýchlostnou sieťovou kartou. Ako sondy v monitorovacej architektúre sa občas používajú aj prvky funkčnej infraštruktúry siete podporujúce potrebnú monitorovaciu funkcionalitu. Sond býva v monitorovanej sieti umiestňovaných viacero a na rôznych miestach.

Komunikačný protokol na export záznamov o tokoch zo sondy na kolektor (červené šípky). Existuje viacero štandardne používaných protokolov realizujúcich popísanú funkcionalitu. Najpoužívanejšími z nich sú protokoly NetFlow a IPFIX. Štandardne je v rámci monitorovacej architektúry nasadený jednotný komunikačný protokol, ale nič nebráni súčasnému použitiu viacerých rôznych protokolov.

Kolektor zbiera a ukladá záznamy o tokoch získavané sondami. Pri veľkom počte sond nasadených na vysokorýchlostných linkách je objem dát tečúcich na kolektor relatívne veľký. Realizovať ho preto treba strojom s dostatočným výpočtovým výkonom a hlavne úložným priestorom. V monitorovacej architektúre je možné použiť jeden aj viacero kolektorov.

Analyzátor sa stará o dodatočné spracovanie dát z kolektoru a ich zobrazenie vo vhodnej podobe užívateľom. Môže navyše realizovať automatickú alebo poloautomatickú pokročilejšiu analýzu nameraných dát s cieľom hľadania neštandardných hodnôt prípadne zaujímavých rysov. Spojenie kolektora s užívateľmi je najčastejšie realizované formou istého webového rozhrania. Takisto je možné funkčnosť analyzátora realizovať priamo na kolektore.

Jednotlivé prvky monitorovacej architektúry je zvyčajne možné vzdialene konfigurovať podľa aktuálnych potrieb monitorovania. Pre potreby tejto práce je najzaujímavejším z prvkov monitorovacej architektúry sonda, preto je jej funkčnosť bližšie rozobraná v nasledujúcom odseku.

Monitorovacia sonda (ang. probe) je niekedy označovaná tiež ako exportér záznamov o tokoch (ang. flow exporter). Hardvérové vybavenie sondy musí poskytovať dostatočný výkon na zachytenie a spracovanie všetkých paketov tečúcich sledovaným bodom. Spracovanie každého paketu začína jeho analýzou a extrahovaním informácií potrebných na jeho jednoznačné priradenie do toku. Na základe extrahovaných informácií je vykonané vyhľadanie v tabuľke aktívnych tokov. V prípade, že záznam o toku ešte neexistuje je v tabuľke aktívnych tokov vytvorený nový záznam pre tok zodpovedajúci spracovávanému paketu. Ak už existuje záznam o aktívnom tok zodpovedajúci spracovávanému paketu, sú informácie v tomto zázname aktualizované. Na správne fungovanie mechanizmu ukončovania tokov popísaného ďalej, je potrebné pakety patriace jednému toku spracovať v poradí, v akom boli spozorované na linke. Prehodenie poradia spracovávania paketov toku môže viesť k nepresnosti v získanej informácii (napr. tok ukončený SYN príznakom skôr ako sú spracované všetky jeho pakety). Tento problém nás obmedzuje hlavne pri snahe rozdeliť spracovanie paketov na viac vlákien.

Okrem zachytenia a spracovania paketov musí na sonde neustále bežať aj proces údržby tabuľky aktívnych tokov. Ten kontroluje stav jednotlivých aktívnych tokov a rozhoduje o vyradzovaní a exporte záznamov skončených tokov. Tok je možné považovať za skončený z nasledujúcich štyroch dôvodov:

Zistený koniec toku: napr. spozorovaním paketu s nastaveným príznakom RST alebo SYN v TCP hlavičke.

Vypršal aktívny timeout: doba trvania toku je zhora časovo ohraničená, ak tok trvá príliš dlho, je považovaný za ukončený aj napriek neustálemu príchodu jemu patriacich paketov. Hodnota aktívneho timeout býva nastavovaná v ráde minút.

Vypršal neaktívny timeout: doba medzi príchodom dvoch po sebe idúcich paketov jedného toku je zhora ohraničená. Neaktívny timeout je nastavovaný na hodnoty v ráde sekúnd.

Preplnenie tabuľky tokov: tabuľka záznamov o tokoch má len konečnú veľkosť, pri prekročení jej kapacity je preto potrebné vhodne vybrať tok, ktorý je umelo ukončený a odstránený z tabuľky, aby bolo možné uložiť záznam pre novo vzniknutý tok.

Export záznamov o tokoch na kolektor nemusí prebehnúť hneď po ich ukončení. Štandardne sa export odkladá tak, aby bolo možné odoslať niekoľko záznamov tokov súčasne v rámci jedného paketu.

Najbežnejšie využívaným protokolom na export dát je v dnešnej dobe protokol NetFlow s najnovšou verziou 9 popísanou v RFC 3954 [14]. Vytvorený a vyvíjaný je firmou Cisco. Napriek tomu, že ide o proprietárny protokol, niektoré jeho verzie sú implementované aj na iných platformách. Najrozšírenejší je dnes vo verzii 5 a najnovšej verzii 9. Stupeň rozšírenia jeho používania umožňuje pokladať ho za istú formu priemyselného štandardu. Z pohľadu podpory exportovaných informácií je verzia 5 dnes už nedostatočná. Umožňuje nasadenie len v IPv4 sieťach pretože nepodporuje IPv6 adresy v záznamoch tokov a zároveň limituje monitorovanie fixným formátom exportovaných záznamov. Takisto je obmedzený podporou len protokolov TCP a UDP na transportnej vrstve a nepodporovaním niektorých rozšírených protokolov iných vrstiev (napr. IPX alebo VLAN). Pre uvedené nevýhody sa v nových zariadeniach odporúča implementovať novšiu verziu NetFlow. Verzia 9 je na tom už podstatne lepšie – podporuje IPv6 adresy v záznamoch a umožňuje istú flexibilitu formátu exportovaných dát pomocou možnosti definície užívateľských šablón formátu exportovaných dát. Aj keď má protokol NetFlow a ním definovaný formát exportovaných dát niekoľko nedokonalostí, je určite veľmi kvalitným a dobre podporovaným riešením exportu záznamov o tokoch.

Snaha definovať nezávislý a štandardný formát exportu tokov viedla ku vzniku iniciatívy, ktorej produktom je návrh na štandard protokolu IPFIX (IP Flow Information Export) [15]. Základom pre IPFIX je NetFlow verzie 9, ktorý je vhodne rozšírený. Vývoj zastrešuje organizácia IETF (Internet Engineering Task Force) a podieľajú sa na ňom aj tvorcovia NetFlow. IPFIX v sebe definuje nielen formát exportu dát, ale aj monitorovaciu architektúru a požadovanú funkčnosť sondy. Definovaný formát exportovaných dát je flexibilný a nastaviteľný užívateľom formou šablón podobne ako v NetFlow verzie 9. IPFIX navyše pre exportované dáta definuje informačný a dátový model, ktoré spolu určujú aké dátové elementy je možné o tokoch sledovať a ako jednotlivé elementy reprezentovať. Svojou flexibilitou a robustnosťou tak IPFIX predstavuje perspektívny návrh na štandard a čoskoro je očakávané masívnejšie rozšírenie jeho podpory a použitia v monitorovacích aplikáciách.

2.4 Bezpečnosť sietí

Nasledujúci text je vytvorený na základe informácií uvedených v dokumentoch [16, 17, 18, 19]. Pripojením zariadenia do internetu je vytvorené fyzické spojenie s niekoľkými tisícmi neznámych sietí a miliónmi neznámych užívateľov. Na jednej strane síce toto spojenie poskytuje jednoduchý prístup k zaujímavým službám a umožňuje rýchle získavanie a zdieľanie informácií. Na druhej strane však predstavuje istú prístupovú cestu k pripojovanému zariadeniu zo strany ostatných užívateľov internetu. Na pripojenom zariadení pritom väčšinou nie sú len dáta a služby, ktoré majú byť prístupné všetkým užívateľom. Vzniká preto problém s nutnosťou existencie možnosti obmedzenia prístupnosti istých informácii a služieb zo siete. Riešením uvedeného problému sa zaoberá práve oblasť bezpečnosti sietí.

Sieťovej bezpečnosti je v dnešnej dobe vďaka neustálemu rastu využitia počítačových sietí prikladaný stále väčší význam a nie je zriedkavé, že správcovia sietí venujú väčšie úsilie bezpečnosti siete ako jej samotnej konfigurácii a správe. Základnú úlohu zabezpečenia siete je možné rozdeliť na dva základné celky. Prvým celkom je zabezpečenie ochrany dôverných informácií pred získaním a úpravou neautorizovanými užívateľmi. Ochranu dôverných informácií je pritom potrebné riešiť na dvoch miestach zároveň. V sieti sa totiž informácie nemusia nachádzať len vo forme dát na disku alebo v pamäti pripojeného koncového zariadenia, ale môžu existovať aj vo forme dát nesených paketmi sieťovej komunikácie tečúcimi aj mimo spravovanú sieť. Druhým celkom je zabezpečenie ochrany siete a zariadení v nej pred zneužitím a útokmi od užívateľov mimo siete. Významnú úlohu pri tom zohráva aj schopnosť včasnej detekcie prebiehajúcich útokov. Okrem útokov vedených proti spravovanej sieti zvonka je niekedy vhodné sledovať a zamedzovať aj snahe o útoky vedené v rámci spravovanej siete alebo smerom z nej.

Zabezpečenie ochrany dôverných informácií pred neautorizovaným zásahom pri prenose sieťou predstavuje významnú časť oblasti sieťovej bezpečnosti. Nutnosť spomenutej ochrany vedie k potrebe zaistenia bezpečnej sieťovej komunikácie medzi koncovými zariadeniami. Komunikácia je považovaná za bezpečnú ak spĺňa tri podmienky [20]: dôvernosť, autenticita a integrita. K uvedenej trojici sa niekedy pridáva ešte požiadavka na splnenia kritéria nepopierateľnosti. Zabezpečenie vlastností potrebných na bezpečnú komunikáciu je v dnešných sieťach postavené na využití princípov modernej kryptografie. Realizuje sa najmä šifrovanie a podpisovanie obsahu prenášaných správ, či certifikácia kľúčov na to potrebných. Podrobnejší rozbor tejto časti sieťovej bezpečnosti nie je pre potreby práce veľmi zaujímavý a nebude ďalej uvádzaný.

Významnou časťou sieťovej bezpečnosti je ochrana siete a jej zariadení pred útokmi a zneužitím. Na správne zaistenie tejto ochrany je veľmi dôležité poznať odkiaľ môžu prichádzať hrozby pre sieť. Potenciálnych útočníkov je možné rozdeliť do niekoľkých základných skupín, kde každá skupina používa trochu odlišné metódy útokov a vyžaduje aj odlišný spôsob obrany. Základnými skupinami útočníkov sú:

Odhodlaný útočník zvonka: snažiaci sa o úmyselný útok práve na danú konkrétnu sieť. Úmyslom býva najčastejšie získanie citlivých informácií zo siete, neautorizovaný prístup k službám siete alebo narušenie správnej obsluhy autorizovaných užívateľov. Problémom ochrany proti odhodlanému útočníkovi je fakt, že bude postupne skúmať stále iné a sofistikovanejšie spôsoby prevedenia útoku proti konkrétnej sieti. Zároveň v prípade úspešného útoku spôsobí cielené a často rozsiahle škody.

Odhodlaný útočník zvnútra: predstavuje osobu vnútri konkrétnej siete, ktorá chce úmyselne zaútočiť na istú jej časť podobne ako útočník zvonka. Výhodou vnútorného úto-

čníka je nepotrebnosť získavania prístupu do siete a často tiež istá vedomosť o fungovaní vnútornej siete a jej zabezpečení. Včasné odhalenie a zamedzenie sieťových útokov zvnútra tak býva často veľmi náročné. Medzi najčastejších útočníkov tohto typu patria nespokojní, bývalí alebo vonkajším útočníkom podplatení zamestnanci organizácií.

Script Kiddy: je termín používaný na označovanie nie príliš sofistikovaných útočníkov, ktorí využívajú ľahko dostupné cudzie metódy na zneužitie dobre známych chýb. Zároveň ich útok nie je cielený na konkrétnu sieť, ale skôr má charakter prechádzania veľkého počtom IP adries v snahe nájsť jednoducho zraniteľný systém. Efektívnou ochranou proti tomuto typu útočníkov je včasná aktualizácia systému a zatváranie známych dier v konfigurácii zariadení siete.

Okrem nutnosti poznať možných pôvodcov útokov je dôležité poznať aj samotné metódy používané pri útokoch. Podobne ako existujú a sú stále zdokonaľované nástroje na ochranu sietí napreduje aj rozvoj rôznych útočných programov a vznikajú stále nové sofistikovanejšie formy útokov. Jednotlivé typy útokov sa od seba navzájom líšia účelom, zložitosťou aj potenciálnou nebezpečnosťou pre sieť a zariadenia v nej. Základné známe metódy používané útočníkmi je možné rozdeliť do nasledujúcich kategórií:

Skenovanie: samo o sebe nepredstavuje útok, ale väčšinou len prípravu naň. Útočník generuje požiadavky na vytvorenie spojenia na náhodné adresy v sieti alebo náhodné porty na zariadení a sleduje, na ktoré z požiadaviek mu príde odpoveď. Vo výsledku teda ide o istú formu prieskumu siete alebo zariadenia v sieti s cieľom nájsť cesty kade je možné následný útok viesť. Rozpoznať skenovanie je možné na základe vysokého počtu požiadaviek na spojenie bez odpovede (na neexistujúce ciele).

DoS útok: má za cieľ zneprístupnenie istej služby užívateľom. Útok sa realizuje generovaním takého veľkého objemu požiadaviek, aby úplne zahltil cieľové zariadenie alebo službu. Útok pritom nemusí byť vedený len z jedného zdroja, ale často je distribuovaný medzi viacero zariadení (DDos). Pri útoku z jedného miesta je možné útočníka spoznať podľa enormne veľkého počtu požiadaviek od rovnakého zdroja, pri distribuovanej variante útoku je ale veľmi ťažké rozlíšiť požiadavky útočníkov od požiadaviek legitímnych užívateľov.

Malvér: predstavuje istý druh programu, ktorý sa samovoľne šíri sieťou pomocou známych bezpečnostných chýb alebo s nevedomou pomocou neskúsených užívateľov. Malvér po infikovaní zariadenia môže na ňom vykonávať rôzny druh činnosti, najčastejšie ide o isté sprístupnenie zariadenia útočníkovi, získanie informácií, poškodenie softvérového vybavenia alebo využitie zariadenia pre ďalšie formy útokov (napr. ako člena DDoS útoku).

Penetrácia: má za cieľ prienik do systému alebo zariadenia, či získanie administrátorských práv na ňom. Vedený je najčastejšie využitím istej chyby v softvérovom vybavení zariadenia, ktorá umožní útočníkovi vložiť do zariadenia svoj program.

Veľké množstvo sieťových útokov je dobre známych a popísaných. Rozpoznávanie je preto možné realizovať hľadaním ich charakteristických prejavov v sieťových dátach. Naproti tomu neustále vzniká veľké množstvo nových útokov, ktorých prejavy sú zatiaľ neznáme. Ich detekcia je tak možná len na základe metód sledovania anomálií v sieťových dátach.

Na zaistenie celkovej bezpečnosti siete pred popísanými útočníkmi a útokmi sú správcami vytvárané a aplikované komplexné bezpečnostné politiky. Kľúčovým prvkom celkovej obrany siete je vytvorenie dostatočne kvalitného zabezpečenia na hraniciach spravovanej siete, ktoré zamedzuje politikou rozpoznaným neoprávneným a nebezpečným aktivitám pochádzajúcim z okolia siete. Vzniká tým isté oddelenie vnútra siete (obyčajne istým spôsobom dôveryhodného) od neznámeho alebo nedôveryhodného okolia. Zabezpečenie hraníc siete sa komplexne označuje ako vytvorenie obvodu siete (ang. network perimeter), ktorý často pozostáva z viacerých vrstiev spájajúcich niekoľko spôsobov obmedzenia prístupu k sieti a sledovania podozrivého chovania v prichádzajúcich a odchádzajúcich komunikáciách. Vytvorenie kvalitne zabezpečeného obvodu siete vedie k výraznému zníženiu nebezpečnosti vonkajších hrozieb pre sieť. Vôbec však nerieši vnútorné hrozby a situácie, kedy útočník získa priamy prístup do siete (napr. fyzickým prístupom k zariadeniu v siete).

Dobre vytvorené a robustné zabezpečenie siete obsahuje okrem kvalitnej obvodovej ochrany aj prvky na dosiahnutie vnútornej (hĺbkovej) bezpečnosti. Je možné prirovnať ho k cibuli, ktorá aj po ošúpaní vonkajšej vrstvy stále obsahuje veľa ďalších vrstiev. Zavedenie vnútornej bezpečnosti prináša hlavne odolnosť voči útokom vedeným zvnútra siete a obmedzuje možnosti vonkajšieho útočníka po úspešnom prekonaní obvodu siete. Realizuje sa často vzájomným oddelením prvkov vnútornej siete a ich združením do istých menších chránených podsietí. Teda akoby zavedením istej hierarchie v tvorbe sieťových obvodov. Dôležité pre hĺbkovú bezpečnosť je aj bezpečnosť jednotlivých zariadení v sieti a poučenie užívateľov o zásadách bezpečnej práce v sieti. Zabezpečeniu zariadení napomáha využitie rôznych antivírových programov, aktualizovanie softvéru alebo pravidelné audity ich stavu.

Pri snahe o čo najlepšiu celkovú bezpečnosť siete je možné využívať niektoré bezpečnostné funkcie poskytované priamo zariadeniami jej infraštruktúry. Infraštruktúrne zariadenia však sami o sebe často nepostačujú na vytvorenie kvalitného zabezpečenia. Do sietí je preto potrebné nasadzovať rôzne typy špecifických bezpečnostných zariadení a aplikácií. Medzi najbežnejšie používané z nich patrí firewall, systém detekcie narušenia (IDS) a systém prevencie narušenia (IPS). Podrobnosti funkcie týchto zariadení sú popísané v nasledujúcom texte.

Firewall je zariadenie starajúce sa o ochranu prístupu k istej časti siete podľa správcom nastavených vlastností. Ochranu realizuje na základe súboru pravidiel určujúcich, ktorej sieťovej komunikácii umožniť priechod a ktorú naopak zablokovať. Základ funkčnosti firewallu je tvorený filtrovaním paketov tečúcich cezeň. Koncept filtrovania paketov spočíva v určovaní, či paketu povoliť vstup do siete resp. výstup zo siete na základe jeho vlastností, ktoré sú určované hlavne využitím informácií uvedených v hlavičkách paketu. Najčastejšie sú používané políčka z hlavičky protokolov sieťovej (IP) a transportnej vrstvy (TCP a UDP). Konkrétne ide hlavne o rozhodovanie na základe IP adries, čísel portov alebo použitého transportného protokolu. Existujú a bežne sa používajú rôzne typy firewallov. Konkrétne ide o statické, stavové a proxy firewally. Statický firewall využíva filtrovanie paketov na základe dopredu pevne danej množiny klasifikačných pravidiel. Stavový firewall analyzuje pakety podrobnejšie a uchováva navyše informácie o aktuálnom stave vytvorených a prebiehajúcich komunikácií. Na základe stavov spojení je schopný dynamicky meniť úroveň inšpekcie paketov (podrobne stačí preskúmať len prvý paket spojenia) a používanú množinu klasifikačných pravidiel. Celkovo tak umožňuje kvalitnejšie filtrovanie paketov. Napokon proxy firewall pracuje na trochu odlišnom princípe ako predošlé dva typy. Na rozdiel od nich nerozhoduje o prepúšťaní cez neho tečúcich paketov komunikácií, ale predstavuje istú aktívnu spojku (proxy server) medzi vnútornou a vonkajšou sieťou. Komunikujúci počítač tak komunikuje len s proxy serverom, ktorý za neho ďalej komunikuje so zamýšľaným

cieľom. Výhodou tohto prístupu je znemožnenie priamych spojení medzi vnútornou a vonkajšou sieťou.

Hlavnou úlohou systémov detekcie narušení (IDS, Intrusion Detection System) je odhaľovanie nežiadaných a potenciálne nebezpečných udalostí na sieti a upozorňovanie správcu na ne. IDS sa používajú na dvoch miestach siete. Buď sa nachádzajú na konkrétnom jednom zariadení a monitorujú jeho správanie alebo odchytávajú a monitorujú komunikácie prebiehajúce na sietí a hľadajú v nich podozrivé aktivity. Po zaznamenaní nechcenej udalosti na sieti je možné upozorňovať na ňu rôznymi spôsobmi ako je zaslanie emailu alebo zápis do logu. Detekcia a upozorňovanie od IDS umožňuje správcovi identifikovať a správne reagovať na aktuálne hrozby sieti. Všeobecne je možné funkcionalitu IDS špecifikovať ako snahu hľadať preddefinované signatúry škodlivých udalostí na základe štatistickej analýzy alebo analýzy anomálií v správaní. Signatúra z pohľadu sieťových IDS predstavuje istý vzor správania sa zodpovedajúci útoku alebo výskytu nežiadanej udalosti na sieti. Tento vzor pritom môže predstavovať jednoduchú vlastnosť jedného paketu, celej komunikácie alebo viacerých súčasných či po sebe idúcich komunikácii. Detekcia na základe spoločných signatúr nad celou sieťovou komunikáciou môže viesť k veľkému počtu falošných detekcií, často je preto využívané aplikovanie istých signatúr len na špecifický typ komunikácie. Napríklad signatúry útokov na DNS služby nie je potrebné hľadať v emailovej komunikácii. Princíp analýzy sieťových anomálií spočíva v prvotnom ustanovení pravidiel normálneho fungovania siete v priebehu času a následnej detekcii významných odchýlok aktuálneho stavu siete od normálneho správania sa. Pozorované anomálie môžu potom z pohľadu IDS predstavovať samostatné hrozby alebo slúžiť ako časti signatúrnych vzorov útokov.

Systémy prevencie narušenia (IPS, Intrusion Prevention System) pridávajú k detekcii hrozieb poskytovanej IDS navyše aj vrstvu automatizovanej aktívnej obrany siete pred rozpoznanými hrozbami bez nutnosti priameho zásahu administrátora. Automatizovaná okamžitá obrana vedie umožňuje zníženie škôd spôsobených útokom. Funkcionalitu IPS je možné chápať ako isté spojenie IDS a firewallu umožňujúce blokovanie paketov podozrivých komunikácií na základe detekcií IDS.

2.5 Softvérovo definované sieťovanie

Text tejto sekcie vychádza z dokumentu [21]. Software Defined Networking (SDN) je novo vznikajúci pohľad na stavbu architektúry počítačových sietí, ktorý sa v poslednej dobe teší stále rastúcej popularite. Vznikol a vyvíja sa na základe spoločného úsilia viacerých komerčných firiem vedených združením Open Networking Foundation (ONF). ONF predstavuje neziskové priemyselné združenie s úlohou zohrávať hlavnú rolu vo vývoji a štandardizovaní dôležitých elementov SDN architektúry. Jedným z prvých úspechov ONF je vytvorenie štandardu protokolu OpenFlow predstavujúceho prvé jednotné komunikačné rozhranie vytvorené špeciálne pre potreby SDN.

Hlavnou motiváciou za vznikom SDN je stále nutnejšia potreba novej a lepšie prispôsobenej realizácie sieťovej architektúry dnešnému spôsobu využitia sietí. Štandardný model sieťovej infraštruktúry je optimalizovaný hlavne na podporu komunikácií typu klient-server. V dnešnej dobe je však stále viac oblastí využitia sietí, kde tento typ komunikácií nie je dominantným. Ide napríklad o dátové centrá alebo distribuované výpočtové systémy. Medzi základné trendy dnešných sietí vedúce k nutnosti reformovať sieťový model patria napríklad:

Zmena vzorov použitia sietí: ako už bolo spomenuté vyššie, model klient-server sa v niektorých oblastiach sieťovania využíva stále menej často.

Používanie mobilných zariadení: jeden užívateľ často pristupuje do siete z rôznych bodov a pomocou rôznych osobných mobilných zariadení ako sú mobilné telefóny či tablety.

Spracovanie veľkých objemov dát: výpočtovo a dátovo náročné úlohy sú riešené aplikovaním masívneho paralelizmu pomocou veľkého počtu výkonných serverov potrebujúcich vzájomnú komunikáciu typu každý s každým.

Pri použití štandardnej architektúry sieťovej infraštruktúry je v dnešnej dobe stále častejšie narážané na problémy, ktoré obmedzujú efektívnu využiteľnosť sietí. Medzi základné problémy patria napríklad:

Komplexnosť vedúca k statickosti: zložitosť vykonanie akejkoľvek zmeny v komplexnej sieti vedie správcom k snahe minimalizovať zásahy do fungujúcej siete. Tento stav výrazne kontrastuje s dynamickou povahou sietí v dnešnej dobe.

Nekonzistencia politík: pri snahe vytvoriť a spravovať konzistentnú politiku v rámci siete sa naráža na problém nutnosti zmeny konfigurácie veľkého počtu zariadení rôznych výrobcov po celej sieti. Pre veľké siete je táto úloha skoro neriešiteľnou.

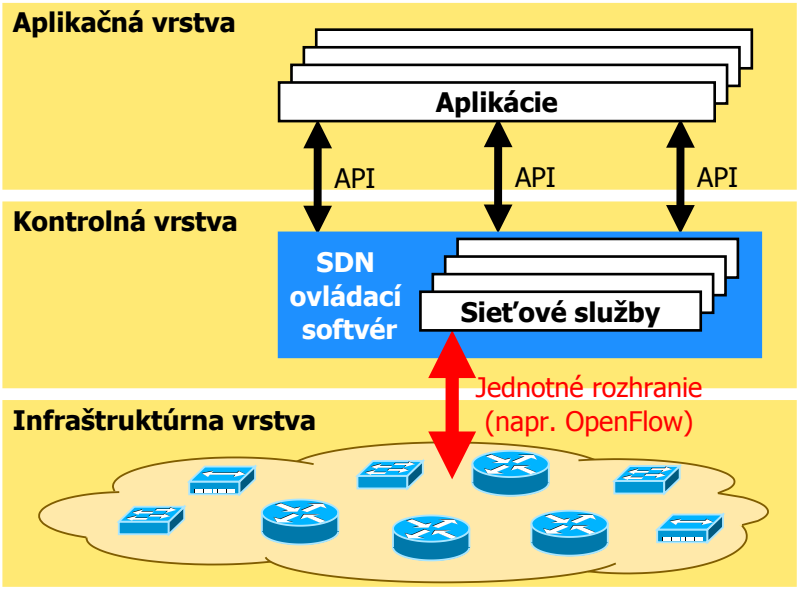
Slabá rozšíriteľnosť: s rastom veľkosti sieti rastie neúmerne komplexnosť a zložitosť jej správy. Správa veľkých sietí sa tak stáva veľmi komplikovanou.

Závislosť na výrobcoch: pri snahe nasadiť novo vzniknutú službu do siete je nutné čakať na začlenenie jej podpory do sieťových zariadení ich výrobcami.

Pri tvorbe SDN sa na uvedené problémy berie ohľad a sú pri jeho návrhu vhodne riešené.

Hlavnou myšlienkou navrhovanej architektúry siete v rámci SDN je striktné oddelenie kontroly a riadenia (inteligencie) sieťovej infraštruktúry od jej funkčných prvkov starajúcich sa o smerovanie dát (substrátu). Zároveň je možné riadenie siete priamo programovať. Myšlienku uvedeného rozdelenia funkčnosti je možné prirovnať fungovaniu procesorových jadier (substrátu) riadenému inštrukciami programu (inteligencie) v dnešných počítačoch. Rozdelenie kontrolnej a dátovej časti umožňuje zavedenie istej formy abstrakcie nad funkčnou infraštruktúrou. Z pohľadu aplikácií a sieťových služieb je tak možné pozerať sa na sieť ako na istú jednotnú logickú alebo virtuálnu entitu poskytujúce spojenie koncových zariadení. Popísaná abstrakcia prináša veľmi jednoduché ovládanie a automatizáciu fungovania sieťovej infraštruktúry umožňujúc tak jednoduché tvorenie škálovateľných a flexibilných sietí.

Z logického pohľadu je možné architektúru SDN rozdeliť do niekoľkých vrstiev ilustrovaných na obrázku 2.10. Nad fyzickými prvkami tvoriacimi infraštruktúru siete v spodnej časti obrázku je vytvorená abstraktná kontrolná vrstva. V rámci kontrolnej vrstvy je z logického pohľadu inteligencia siete centralizovaná v softvérových SDN radičoch (kontrolóroch), ktoré udržujú a spravujú celkový globálny pohľad na sieť. Výsledkom zavedenia centralizovane kontrolnej vrstvy sa aplikáciám (aplikačná vrstva) javí sieť ako jediný veľký logický prepínač. SDN tak je schopné umožniť kontrolu celej siete z jediného logického miesta a navyše nezávisle na špecifických vlastnostiach jednotlivých prvkov a služieb tvoriacich jej infraštruktúru. Popísané odtienenie špecifických vlastností prvkov tak vedie k výraznému zjednodušeniu návrhu a správy sietí. Z druhej strany zase zavedenie jednotného riadiaceho rozhrania SDN umožňuje zjednodušenie aj samotných prvkov infraštruktúry, ktorých jedinou úlohou je prijímať a vykonávať inštrukcie od SDN kontrolórov.



Obr. 2.10: Základný pohľad na architektúru SDN

Centralizácia riadenia sieťovej infraštruktúry zavedená v logickej architektúre SDN umožňuje jednoduchú podporu pre programové riadenie siete, kedy správcom stačí definovať a upravovať parametre chovania siete na jednom mieste. Naproti tomu, v štandardnom chápaní dnešných sietí sú parametre ovplyvňujúce chovanie siete roztrúsené po niekoľkých rôznych zariadeniach infraštruktúry a ich spravovanie je najmä pre veľké siete veľmi pracné. Ovládanie chovania siete z jediného miesta môže výrazne urýchliť nasadzovanie nových aplikácií a služieb do siete. Centralizáciou kontrolnej vrstvy je tiež umožnená flexibilita správy, optimalizácie a zabezpečenia sieťových prvkov pomocou automatizovaných SDN programov. Navyše je možné novú funkcionalitu siete docieliť vlastným tvorením SDN programov bez nutnosti čakať na zaradenie podpory tejto funkcionality výrobcom do zariadení v infraštruktúre.

Okrem samotnej abstrakcie nad sieťou, počíta architektúra SDN aj s podporou istej množiny aplikačných rozhraní (API). Tieto rozhrania slúžia na umožnenie implementovania štandardných sieťových služieb ako sú smerovanie, multicast, riadenie prístupu, kvalita služieb, bezpečnosť a podobne, podľa potrieb danej siete. Zároveň je nasadenie niektorých služieb zjednodušené, vďaka centralizácii vedúcej k jednoduchému zaisteniu konzistentnej politiky danej služby v rámci celej siete. Použitie vhodných rozhraní medzi SDN kontrolórmi a užívateľskými aplikáciami je možné docieliť využitie služieb siete bez nutnosti podriadenia sa špecifickým detailom jej implementácie. SDN tak umožní pozerať sa na siete ako na systémy poskytujúce aplikačne špecifickú funkcionalitu. Podobne pri definovaní aplikácií bude stačiť rozmýšľať len o umožnení/neumožnení sieťovej komunikácie namiesto nutnosti riešenia konkrétnych špecifík tejto komunikácie.

Snaha vyvíjaná okolo SDN združením ONF viedla k dnešnému dňu k vzniku a zlepšovanie protokolu OpenFlow. OpenFlow je prvým štandardným komunikačným rozhraním definovaným medzi kontrolnou a infraštruktúrnou vrstvou SDN architektúry. Úlohou OpenFlow je teda umožniť priamy a jednotný prístup k zariadeniam sieťovej infraštruktúry zaisťujúcim smerovanie dát, akými sú napríklad smerovače a prepínače. Jednotný prístup je zabezpečený k fyzickým aj virtuálnym zariadeniam rovnako ako zariadeniam rôznych

výrobcov. Žiaden iný štandardný protokol v dnešnej dobe neposkytuje jednotný prístup k zariadeniam siete podobne ako OpenFlow. Zároveň práve táto jednotnosť je základom pre architektúru SDN sietí.

Protokol OpenFlow špecifikuje základné primitíva používané na vzdialenú kontrolu smerovania vykonávaného v zariadeniach sieťovej infraštruktúry. Podporovaný musí byť jednak samotnými kontrolovanými zariadeniami, ale aj SDN kontrolórmi. Na identifikáciu sieťovej premávky a jej riadenie používa OpenFlow princíp založený na tokoch predstavených už v predošlom texte (sekcia 2.3). Tento princíp pracuje s množinou dopredu definovaných statických a dynamických pravidiel nad jednotlivými tokmi alebo ich skupinami nastavovanými SDN kontrolným softvérom. Podpora definovania funkčnosti na jednotlivých tokoch umožňuje dosiahnutie jemnej kontroly chovania SDN siete a tým jej umožňuje vhodné prispôsobenie sa na aktuálne prebiehajúcu situáciu v reálnom čase pre konkrétne aplikácie, užívateľov a komunikácie. Štandardné IP siete túto možnosť kontroly neposkytujú, keďže smerovanie paketov všetkých komunikácií medzi dvomi koncovými zariadeniami je v nich riadené rovnakými pravidlami.

Protokol OpenFlow bol pôvodne vytvorený len pre Ethernet siete, avšak jeho použiteľnosť je dnes ďaleko širšia. SDN pracujúce s OpenFlow môže byť a v dnešnej dobe aj je nasadzovaný (napr. firmou Google) na už existujúcich fyzických a virtuálnych sieťach. Použiť je ho možné paralelne s už existujúcim tradičným smerovaním, čo umožňuje správcom jednoduchý postupný prechod na SDN technológiu a použitie SDN aj v sieťach kde nie všetky zariadenia podporujú OpenFlow. Zároveň umožňuje nasadenie SDN obmedziť len na tie časti siete, kde je jeho nasadenie prínosným.

2.6 Existujúca hardvérová akcelerácia – Hanic

Popis platformy Hanic uvedený v tejto sekcii je postavený na verejne dostupných dokumentoch [22, 1]. Používané označenie Hanic predstavuje skratku z celého názvu, ktorý sa používa v dvoch verziách: Hardware Accelerated Network Interface Card (hardvérovo akcelerovaná sieťová karta) alebo HAshing Network Interface Card (hašovacia sieťová karta). Platforma Hanic predstavuje súbor softvéru a firmvéru, ktorý prináša pokročilejšiu funkcionalitu na hardvérovo akcelerované sieťové karty rodiny COMBOv2. Hanic podporuje spracovanie paketov na plnej rýchlosti pripojených liniek a ich prenos do softvéru cez PCI express zbernicu pomocou DMA prenosov. V súčasnosti podporuje Hanic rýchlosti liniek do 10 Gb/s a najrozšírenejšie používanou je verzia s dvomi sieťovými rozhraniami rýchlosti 10 Gb/s, ktorá je využívaná napríklad na monitorovanie siete združenia Cesnet.

Hlavnou podporovanou funkcionalitou platformy Hanic je schopnosť delenia prijatého vstupného toku dát zo siete medzi niekoľko softvérových rozhraní. Dáta na každom z týchto rozhraní môžu byť následne spracované na inom procesorovom jadre a teda záťaž spojená so spracovaním sieťových dát tak efektívne rozdelená medzi viacero jadier procesoru. Hanic podporuje rozdeľovanie paketov do softvérových kanálov na základe konfigurovateľnej haš hodnoty vypočítanej z vybraných položiek hlavičiek paketov. Popísaná distribúcia má veľkú výhodu v tom, že všetky pakety patriace do rovnakého sieťového toku sú zaradené do rovnakého kanálu a tým spracovávané rovnakým jadrom. Výpočet haš hodnoty na priradenie paketov do kanálu je možné konfigurovať a realizovať na základe ľubovoľnej podmnožiny nasledujúcich hodnôt z hlavičiek:

Verzia IP: hodnota políčka Version z IPv4 alebo IPv6 hlavičky.

Zdrojová a cieľová IP adresa: pre IPv4 aj IPv6 protokol.

Protokol transportnej vrstvy: získavaný z IPv4 alebo IPv6 hlavičky.

Číslo zdrojového a cieľového portu: z hlavičiek protokolov TCP a UDP.

Hanic okrem vyššie popísanej hlavnej funkcionality poskytuje podporu aj pre niekoľko ďalších rozširujúcich funkcií:

Vzorkovanie: podporované je jednak konštantné vzorkovanie kedy je spracovávaný len presne každý n-tý prijatý paket, kde n je konfigurovateľné celé číslo. Druhou formou je vzorkovanie založené na haš hodnote počítanej z paketov, kedy len pakety s haš hodnotou z definovaného rozsahu sú spracovávané.

Distribúcia na kanály: okrem vyššie popísanej haš distribúcie je možné nastaviť aj distribúciu formou Round Robin, teda pakety sú rovnomerne delené do všetkých kanálov nezávisle na tokoch.

Skracovanie: na pevne definovanú maximálnu dĺžku.

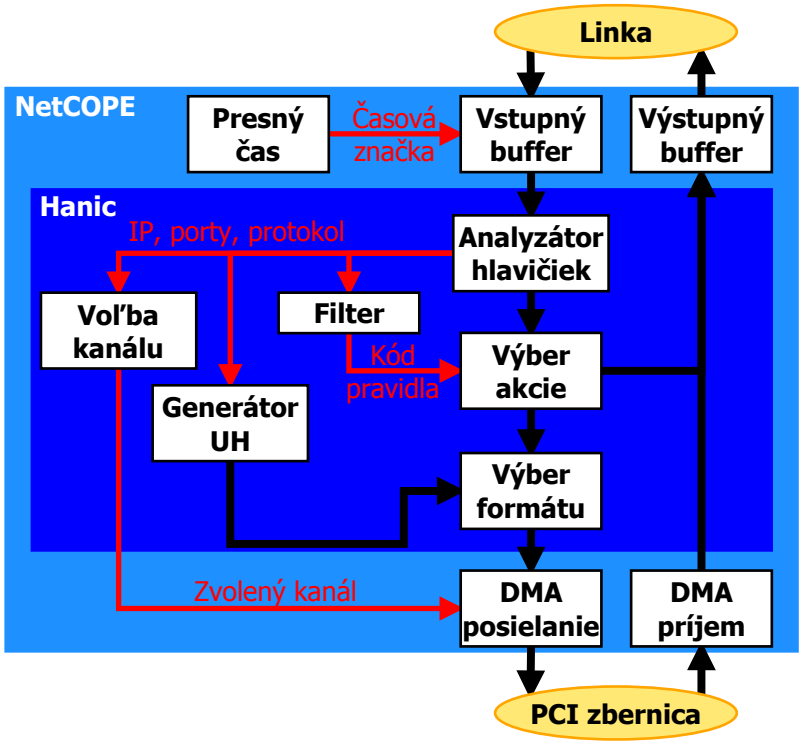
Časové pečiatky: prideľované paketom pri ich zachytení hardvérovou kartou s presnosťou v nanosekundách.

Jednoduchá klasifikácia: pomocou paketového filtra s konfigurovateľnými pravidlami a definovanými akciami pre jednotlivé skupiny paketov. Akciou je voliteľné skrátenie paketu a jeho nasmerovanie do SW, na sieťové rozhranie alebo zahodenie.

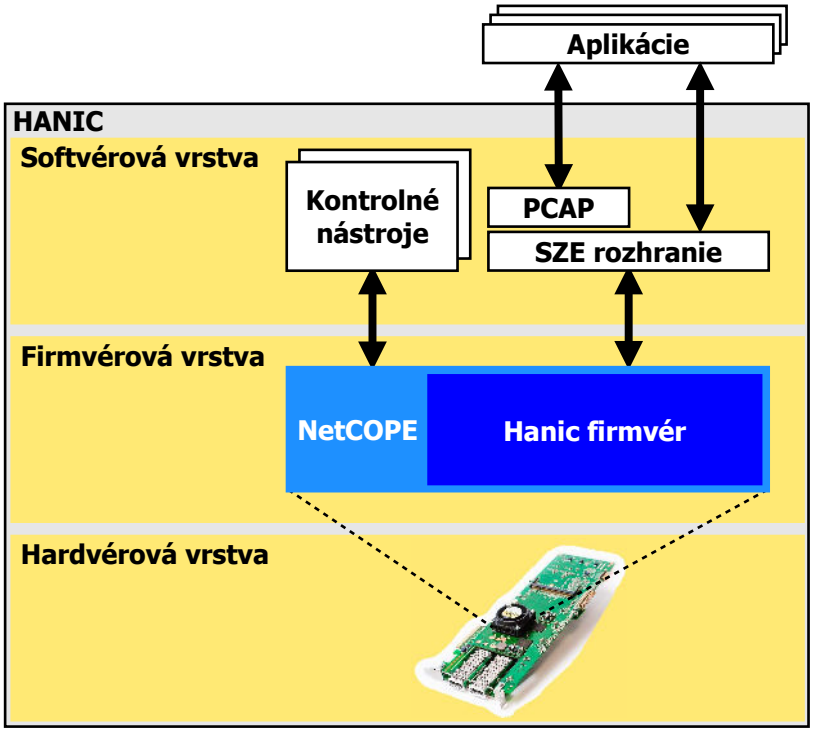
Unifikované hlavičky (UH): namiesto celých prijatých paketov môžu byť do SW kanálov posielané len informácie extrahované z hlavičiek paketov v unifikovanej podobe. Ide o rovnaké informácie ako sú použité na výpočet haš delenia do kanálov.

Softvérová časť Hanic je tvorená súborom niekoľkých skriptov, programov a modulov pre operačný systém Linux. Ich úlohou je hlavne konfigurácia parametrov podporovanej funkčnosti firmvéru Hanic a riadenie či sledovanie jeho behu. Súčasťou sú tiež moduly potrebné na príjem a čítanie paketov zo softvérových kanálov. Hanic podporuje dve možnosti príjmu paketov zo softvéru. Jednou možnosťou je použitie proprietárneho SZE rozhrania, ktoré pred každý rámec vo forme ako bol prijatý zo siete vkladá SZE hlavičku obsahujúcu napríklad časovú pečiatku času prijatia alebo číslo vstupného rozhrania. Druhou možnosťou je využitie podpory SZE v libpcap knižnici. Táto podpora umožňuje prístup k dátam v rovnakej podobe ako zo štandardnej sieťovej karty a umožňuje tak použitie viacerých bežne dostupných aplikácií (wireshark, tcpdump) a tiež použitie štandardného PCAP rozhrania na tvorbu vlastných aplikácií.

Firmvér zariadenia Hanic je určený pre FPGA a pracuje nad platformou NetCOPE. Táto platforma vytvára jednotné a jednoducho uchopiteľné rozhranie nad samotnými hardvérovými prvkami sieťovej karty, na ktorom firmvér beží. Zároveň sa stará o základné operácie potrebné na korektný príjem a odosielanie paketov na linku, priradzovanie časových pečiatok a aj komunikáciu cez PCI zbernicu so softvérom počítača. Zjednodušená schéma základného fungovania firmvéru Hanic je zachytená na obrázku 2.11. Čierne šípky zachycujú trasu rámcov, červené šipky zobrazujú popísaný špecifický typ informácie. Z obrázka je tiež viditeľné, ktorú časť funkcionality realizuje NetCOPE a ktorú samotný Hanic. Po úspešnom prijatí a priradení časových značiek vo vstupnom bufferi sú z hlavičiek rámcov extrahované zaujímavé položky (IP adresy, porty, protokol). Na ich základe je najprv podľa nastavených filtračných pravidiel rozhodnuté kam bude daný rámec ďalej preposlaný.



Obr. 2.11: Zjednodušená schéma fungovania firmvéru Hanic



Obr. 2.12: Vrstvový model súčastí akceleračnej platformy Hanic

Môže smerovať buď späť na linku alebo do jedného zo softvérových kanálov, prípadne môže byť úplne zahodený či len skrátený na definovanú dĺžku. V prípade, že rámec smeruje do softvéru je najprv rozhodnuté, či tam bude poslaný celý jeho obsah alebo len z neho vytvorená UH. Konkrétny softvérový (DMA) kanál je následne vybraný na základe haš hodnoty extrahovaných políčok alebo Round Robin princípom.

Jednotlivé súčasti tvoriace zariadenie Hanic na akceleráciu sieťových aplikácií je možné rozdeliť do vrstiev ako na obrázku 2.12. Samotným funkčným hardvérom je COMBO karta s FPGA čipom. Na nej beží firmvérová časť Hanic nad NetCOPE platformou. Nad firmvérom pracujú základné nástroje na jeho konfiguráciu a prenos sieťových dát medzi kartou a softvérom. Na vrchole, využívajúc funkcionalitu popísaných vrstiev, fungujú samotné užívateľské aplikácie realizujúce požadovanú funkcionalitu akou je napríklad monitorovanie alebo bezpečnostná analýza sieťových tokov.

Podobnú konceptuálnu štruktúru akú má popísaný systém Hanic vo veľkej miere využívajú aj ďalšie hardvérovo akcelerované platformy určené pre požitie vo vysokorýchlostných sieťach a aplikácie nad nimi fungujúce. Za zmienku z nich stojí FlowMon sonda [23] fungujúca na COMBO kartách s platformou NetCOPE alebo NetFPGA kartách. Ide o hardvérovo akcelerovanú NetFlow sondu určenú pre použitie v 1 a 10 Gb/s sieťach. Hlavným zaujímavým rysom FlowMon sondy je riešenie všetkých operácií spojených s monitorovaním sieťových tokov priamo v rámci firmvérovej časti systému. Do softvéru sú tak, na rozdiel od dát jednotlivých paketov ako v prípade Hanic, posielané len výsledné agregované záznamy o monitorovaných tokoch. Úlohou softvéru FlowMon sondy je potom len exportovanie záznamov tokov na kolektor.

Kapitola 3

Konceptuálny návrh

Kapitola sa zaoberá základným návrhom funkcionality systému softvérovo riadenej akcelerácie monitorovacích a bezpečnostných sieťových aplikácií. Návrh systému je inšpirovaný teoretickými znalosťami uvedenými v predošlej kapitole. Ich podrobnejšiemu rozboru z pohľadu potrieb systému sa venuje prvá časť kapitoly. Samotný vytvorený návrh systému je popísaný v druhej časti kapitoly. Systém je inšpirovaný niektorými myšlienkami Software Defined Networking vhodne využitými aby umožnili podporu základných potrieb monitorovacích a bezpečnostných úloh. Preto je pre systém použité označenie Software Defined Monitoring (SDM, softvérovo definované monitorovanie). Počas celého návrhu SDM systému sú zohľadňované výkonnostné parametre potrebné pre nasadenie v 100 Gb/s sieťach a tiež podpora oboch rozšírených verzií protokolu IP (IPv4 aj IPv6).

3.1 Rozbor využiteľných vlastností

Základ všetkých aplikácií z oblasti monitorovania a bezpečnosti sietí tvorí istá forma spracovania dátového toku zo siete s cieľom extrakcie zaujímavých informácií rôzneho typu pre ďalšie spracovanie. S rastúcou rýchlosťou sieťových liniek sa práve toto spracovanie dát stáva kritickým úzkym miestom v návrhu aplikácií. Preto sa hardvérovou akceleráciou podporuje hlavne výkonnosť tejto úlohy. V dnešných 1 a 10 Gb/s sieťach sa najčastejšie využívajú dva základné modely akcelerácie spracovania dát. Ich použitie pre potreby 100 Gb/s sietí však nie je príliš efektívne.

Prvý z dvojice modelov je založený na využití špecializovaného hardvérového zariadenia (architektúry) určeného na riešenie konkrétnej úlohy. Využité je pritom aplikačne úzko špecializované hardvérové riešenie akcelerácie spracovania sieťových dát. Príkladom tohto prístupu môže byť FlowMon sonda akcelerujúca úlohu monitorovanie sieťových tokov. Hlavnou výhodou špecializovaného hardvéru je jeho vysoká výkonnosť prameniaca z akcelerácie veľkej časti úlohy. Problém však predstavuje nízka flexibilita a vysoká cena tohto prístupu. Riešenie odlišných úloh totiž vyžaduje použitie odlišného špecializovaného hardvérového zariadenia resp. zavedenie výrazných zmien do existujúcej hardvérovej architektúry. V prípade komplexnejších úloh akými sú napríklad spracovanie aplikačných protokolov alebo detekcia útokov môže byť problémom aj vysoká náročnosť prípadne až neexistencia vhodnej hardvérovej realizácie ich riešenia. Všeobecne platí, že vytvorenie istej aplikácie v hardvéry je niekoľkonásobne náročnejšie ako v softvéri.

Druhý rozšírený model akcelerácie spracovania dát je postavený na využití hardvérovo akcelerovaných sieťových kariet. Tieto karty sú väčšinou pripojené v hostiteľskom počí-

tači cez PCI Express zbernicu a akcelerujú len záchyt paketov z linky, ich najzákladnejšie spracovanie (odbalenie L1 vrstvy, kontrola CRC, časové značky . . .) a prenos do pamäte hostiteľského počítača. Na hostiteľskom počítači potom celé spracovanie paketov prebieha v procesoroch kde bežia jednotlivé monitorovacie a bezpečnostné aplikácie. Do popísanej kategórie akcelerácie je možné zaradiť monitorovanie siete združenia Cesnet využitím platforiem Hanic na záchyt paketov. Hlavnou výhodou použitia akcelerovaných sieťových kariet je flexibilita tohto prístupu prameniaca z akcelerácie len veľmi všeobecnej časti spracovania dát. Všetko aplikačne špecifické spracovanie je tak ponechané až na konkrétne softvérové aplikácie, ktoré je možné meniť a upravovať podstatne jednoduchšie ako hardvérové jednotky. Nevýhodou tohto prístupu môže byť všeobecne nižšia výkonnosť, spôsobená potrebou softvérového spracovania celého dátového toku z linky. To pre použitie v 100 Gb/s sieťach vedie k vzniku dvoch veľmi kritických úzkych miest: nedostatočnú výkonnosť procesorov počítača a obmedzenú priepustnosť PCI Express zbernice.

Popísané modely použitia akcelerácie majú závažné nedostatky znemožňujúce ich efektívne nasadenie pri zabezpečovaní úloh monitorovania a bezpečnosti na 100 Gb/s sieťach. Preto je vhodné vytvoriť a používať nový model akcelerácie, ktorý prekoná malú flexibilitu čisto hardvérového a nízku výkonnosť čisto softvérového spracovania dát. Dosiahnuteľné to je vhodným definovaním hardvér/softvér kodizajnu inšpirovaného myšlienkami Software Defined Monitoring. Hardvérová časť navrhovaného systému bude mať schopnosť realizovať rôzne spôsoby predspracovania sieťových dát (funkčný substrát) pričom ponechá všetku kontrolu tohto predspracovania a pokročilejšie časti riešených úloh na softvérovú časť (inteligencia). Riadenie hardvérového predspracovania bude sprostredkované softvérovým kontrolórom (radičom) cez jednoduché rozhranie priamo aplikáciám, ktoré budú realizovať pokročilejšie spracovanie. Podobne ako v OpenFlow, môže radič predspracovanie ovládať nastavovaním pravidiel nad jednotlivými tokmi alebo ich skupinami, čím umožní jemnú kontrolu chovania systému podľa aktuálnej situácie na sieti a potrieb konkrétnych aplikácií. Rozšírenie možností predspracovania dát v hardvéri umožní odľahčiť softvér a zvýšiť tak celkovú výkonnosť. Ponechanie všetkého riadenia a pokročilejšej funkcionality na softvér zároveň zachová vysokú flexibilitu použitia systému.

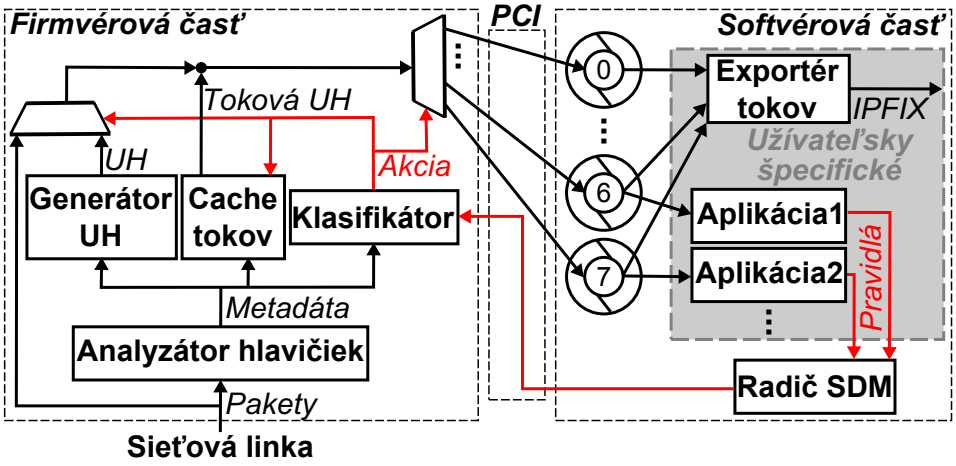
Efektivita využitia popísaného spôsobu akcelerácie veľmi závisí od výberu spôsobov predspracovania dát podporovaných v hardvérovej časti. Vhodnú voľbu je nutné postaviť na základe všeobecnej charakteristiky dátových nárokov monitorovacích a bezpečnostných úloh. Pre základné monitorovanie sú potrebné agregované informácie zo všetkých paketov, ktoré sú získavané hlavne z ich hlavičiek, dátová časť je nepotrebná. Hardvér preto môže poskytovať schopnosť rôznej extrakcie a agregovania zaujímavých dát z paketov, ktoré bude ďalej posielať do softvéru v istej unifikovanej forme namiesto celých paketov. Vhodnými príkladmi takéhoto formátu sú unifikované hlavičky na platforme Hanic a záznamy o tokoch na FlowMon sonde. Výsledkom využitia prenosu len zaujímavých informácií o paketoch je zníženie nárokov na priepustnosť PCI Express zbernice a tiež zníženie vyťaženia procesorov vďaka extrakcii dát realizovanému v hardvéri. Na druhej strane bezpečnostné a pokročilé monitorovacie aplikácie potrebujú potenciálne zaujímavé pakety skúmať viac do hĺbky a tak často zasahujú aj do ich dátovej časti. Pre ich potreby je teda potrebné posielať do softvéru celé pakety vybraných skupín, ostatné je možné zahadzovať. Napríklad detektor útokov cez SSH potrebuje analyzovať len SSH pakety alebo analýza informácií z HTTP hlavičky pracuje len s niekoľkými prvými paketmi každého HTTP toku.

Podporou v predošlom odseku popísaných operácií v hardvéri a jemného riadenia ich použitia je možné docieliť delenie dátového toku zo siete do tried podľa jeho zaujímavosti. Najzaujímavejšiu časť spracovávať celú podrobne v softvéri, menej zaujímavú časť

čiastočne predspracovať v hardvéri a nezaujímavú rovno zahadzovať. Tým je docielené kontrolované hardvérovo akcelerované preriedenie dát sieťového toku resp. kontrolovaná strata nevýznamných informácií. Softvérové aplikácie potom spracúvajú menší objem dát s vyššou informačnou hodnotou, teda využívajú procesorový výkon a prenosové pásmo PCI Express zbernice efektívnejšie. Efektivitu využitia procesorového času na dnes bežných viacjadrových platformách je možné ďalej vylepšiť hardvérovou podporou delenia dát do viacerých komunikačných kanálov (podobne ako Hanic). Toto delenie dát môže byť kontrolovateľné softvérovým radičom podľa potrieb bežiacich aplikácií. Vo výsledku tak navrhovaná platforma bude poskytovať akcelerovanú podporu jemne nastaviteľného delenia a redukcie dátového toku zo siete podľa aktuálnych potrieb rôznych monitorovacích a bezpečnostných aplikácií.

3.2 Software Defined Monitoring

Návrh systému hardvérovej akcelerácie monitorovacích a bezpečnostných aplikácií označeného Software Defined Monitoring vychádza z vlastností popísaných v predošlej sekcii. Základný princíp akcelerácie je teda postavený na softvérovo jemne kontrolovateľnej strate informácií a delení záťaže pri hardvérovom predspracovaní sieťových dát. Riadenie predspracovania je realizované na bázy tokov, teda najmenšou rozlíšiteľnou jednotkou z pohľadu riadenia je sieťový tok. Riadenie predspracovania je realizované výhradne softvérom, preto sú prvé pakety nových tokov posielané na softvérové spracovanie. Na ich základe môžu následne softvérové aplikácie rozhodnúť o využití istého spôsobu predspracovania ďalších paketov daných tokov v hardvéri.



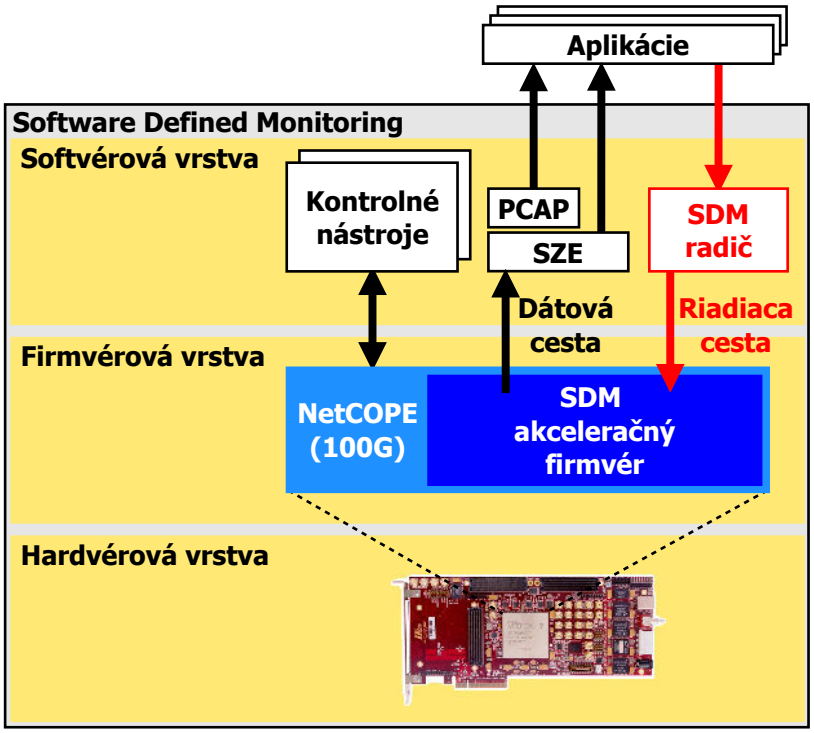
Obr. 3.1: Konceptuálny model návrhu funkcionality SDM

Hlavná schéma navrhovanej funkcionality systému SDM je zobrazená na obrázku 3.1. Ako je možné vidieť, systém sa skladá z firmvérovej a softvérovej časti komunikujúcich spolu cez PCI Express zbernicu. Dátové cesty sú v obrázku zakreslené čiernymi šípkami a riadiace červenými. Spracovanie všetkých paketov prijatých zo sieťovej linky začína analýzou ich hlavičiek a extrakciou zaujímavých metadát (blok Analyzátor hlavičiek). Získané metadáta sú následne použité na klasifikáciu paketu na základe softvérom nastavených pravidiel (blok Klasifikátor). Pravidlá identifikujú konkrétny tok a spôsob zaobchádzania s jeho paketmi v rámci firmvéru, presnejšie: zvolený spôsob predspracovania paketov a číslo im patriaceho softvérového kanálu. Firmvér umožňuje posielanie paketov do softvéru buď bez zmeny alebo

vo forme unifikovanej hlavičky (blok Generátor UH) poskladanej zo zaujímavých dát paketu. Okrem toho umožňuje aj úplné zahadzovanie paketov alebo ich agregovanie do formy záznamov o tokoch (blok Cache tokov), ktoré sú pravidelne exportované do softvéru. Prenos dát do softvéru sa realizuje priamymi zápismi do pamäte cez PCI Express zbernicu a je pri ňom možné deliť dáta do viacerých nezávislých logických kanálov.

Dáta zapisované SDM firmvérom do pamäte sú v nej držané vo forme kruhových zoznamov. Každému logickému kanálu prislúcha práve jeden kruhový zoznam. Softvérové aplikácie, vykonávajúce užívateľom špecifikované úlohy, sú schopné čítať dáta z vybraných kanálov (kruhových zoznamov) cez štandardné operácie práce so sieťovými rozhraniami operačného systému. Dáta im môžu chodiť vo formáte celých paketov, unifikovaných hlavičiek alebo záznamov o tokoch. Okrem príjmu dát sú aplikácie schopné SDM radiču oznamovať svoj záujem resp. nezáujem o dáta jednotlivých tokov. Úlohou SDM radiča je nazbierané informácie od všetkých aplikácií agregovať do formy pravidiel na konfiguráciu predspracovania tokov vo firmvéri. Cieľom je dosiahnutie maximálnej redukciu objemu dát zo siete a maximálneho urýchlenia ich softvérového spracovania.

Trochu iný pohľad na navrhnutý systém SDM ponúka jeho vrstvová schéma zobrazená na obrázku 3.2. Ukázané je, že SDM systém je navrhnutý, podobne ako Hanic zo sekcie 2.6, na fungovanie nad hardvérovou sieťovou kartou osadenou čipom FPGA. Pre potreby zamýšľaného finálneho nasadenia SDM bude použitá karta so 100 Gb/s sieťovým rozhraním a na nej fungujúca verzia NetCOPE alebo jej podobnej platformy. Jadrom firmvéru je SDM akcelerácia spracovania sieťových dát schopná pracovať na plnej rýchlosti linky. Nad firmvérovou vrstvou pracujú základné nástroje na jeho konfiguráciu, prenos sieťových dát do softvéru (čierna dátová cesta) a konfiguráciu SDM firmvéru (červená riadiaca cesta). Na vrchole pracujú samotné aplikácie realizujúce užívateľom požadované monitorovacie a bezpečnostné úlohy.



Obr. 3.2: Vrstvový model súčastí akceleračného systému SDM

3.3 Potenciálne slabé miesta návrhu

Navrhnutý model SDM prináša urýchlenie plynúce hlavne zo softvérom riadeného predspracovania sieťových dát v hardvéri. Jednotlivé formy predspracovania sa istým spôsobom snažia redukovať dátový tok zo siete a znižovať tým záťaž softvéru. Ovládať je ich možné pomocou pravidiel pre jednotlivé toky vkladaných do firmvérovej jednotky v systéme. Spracovanie jednotlivých sieťových tokov začína vždy spracovaním začiatočných paketov v softvéri. Až na ich základe je následne možné konfigurovať hardvérové predspracovanie ďalších paketov toku. Táto vlastnosť zavádza do systému spätnú väzbu začínajúcu klasifikovaním paketov a výberom akcie pre ne, pokračujúcu ich softvérovým spracovaním a končiacu vytvorením a aktivovaním pravidla platného definujúceho predspracovanie ďalších paketov toku. Inak povedané, v systéme existuje isté oneskorenie softvérového výberu akcií na spracovanie nových tokov.

Z pohľadu na koncept SDM systému popísaného v predošlom odseku je možné odvodiť niekoľko potenciálne slabých miest návrhu. Ich existencia môže za veľmi nepriaznivých podmienok viesť na nízku dosahovanú efektivitu hardvérového predspracovania dát a tým nízky stupeň urýchlenia aplikácií. Medzi spomenuté slabé miesta môžu patriť:

Dlhé trvanie spätnej väzby. Pri nasadení vo vysokorýchlostných sieťach nie je možné čakať so spracovaním paketov na výber im zodpovedajúcej akcie na konci softvérovej spätnej väzby, pakety tokov musia byť spracovávané priebežne. Takže zvolená akcia pre tok nedokáže ovplyvniť predspracovanie istej časti jeho úvodných paketov, ktoré sú preto posielané celé do softvéru (ako patriace neznámemu toku). V prípade veľkého počtu extrémne krátko trvajúcich tokov alebo tokov s väčšinou paketov prenášaných tesne po ich začiatku tak dosiahnuteľná efektivita hardvérového predspracovania výrazne klesá.

Obmedzená kapacita firmvéru. Jemnosť riadenia vedie k nutnosti ukladať pravidlo definujúcich aktuálne používané spracovanie ku každému toku. Limitujúca pri tom môže byť kapacita firmvérovej jednotky klasifikovania paketov alebo cache tokov. Extrémne veľké množstvo súčasne prebiehajúcich sieťových tokov na linke môže potom viesť k obmedzeniu spracovania len na istú malú časť z nich. Negatívny dopad uvedeného efektu je okrem dostatočne veľkej kapacity pamäte (napr. použitím externej pamäť na karte) možné eliminovať aj vhodným výberom tokov, ktorých spracovanie má byť akcelerované. Vhodnosť spracovania toku je daná dosiahnuteľnou redukciou jeho dát (efektivitou) pri predspracovaní. Všeobecne je preto vhodné uprednostniť predspracovanie veľkých (ťažkých) tokov.

Nedostatočná redukcia dát. Hardvérové predspracovanie redukuje dátový tok zo siete pomocou tvorby unifikovaných hlavičiek, spracovaním paketov v hardvérovej cache tokov alebo ich zahadzovaním. Prvé dve z uvedených metód dosajú zníženie toku priamoúmerne závislé od veľkosti spracovávaných paketov a tokov. V prípade extrémne krátkych paketov a malých tokov je efektivita redukcie dát uvedenými metódami relatívne nízka.

Vysoká jemnosť kontroly. Voľba základnej jednotky kontroly urýchľovania ovplyvňuje potrebný počet pravidiel v klasifikačnej jednotke aj mieru ich generovania. Pre malé (ľahké) toky je zisk zo zavedenia pravidla na ich akceleráciu relatívne malý. V extrémnom prípade, kedy na sieti prevládajú veľmi ľahké toky, môže byť zisk hardvérovej akcelerácie prekrytý réžiou spojenou s nastavovaním pravidiel. Uvedená situácia môže

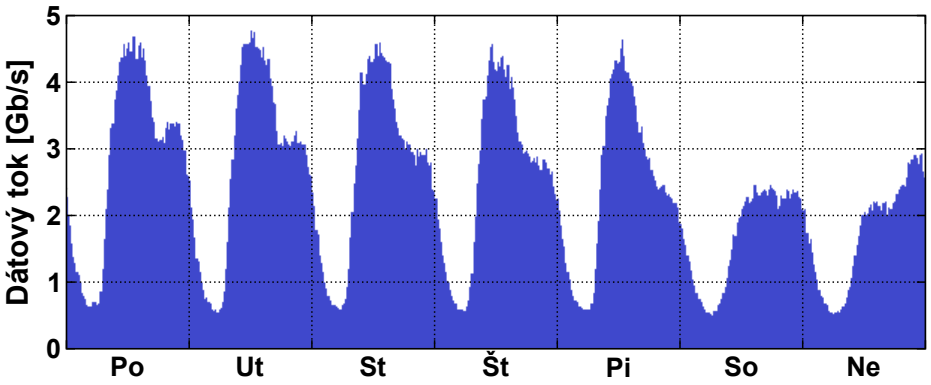
vyústiť aj na privysokú mieru generovania pravidiel oproti priepustnosti konfiguračného rozhrania.

Negatívny dopad všetkých vyššie popísaných problémových miest návrhu SDM úzko súvisí s vlastnosťami spracovávaných sieťových dát. Pred ďalším upresňovaním návrhu je preto vhodné bližšie analyzovať niektoré vlastnosti dát tečúcich na reálnych vysokorýchlostných sieťových linkách. Na základe pozorovaných vlastností bude potom možné odhadnúť veľkosť dopadu jednotlivých kritických miest na celkovú efektivitu systému. Navyše bude možné vykonať prípadnú úpravu častí návrhu SDM systému s cieľom dosiahnutia jeho efektívnejšej funkcionality v reálnom nasadení.

Kapitola 4

Analýza vysokorýchlostného sieťového toku

Kapitola popisuje postup a výsledky meraní rôznych vlastností dátového toku na vysokorýchlostných sieťových linkách. Testovanie prebiehalo v sieti združenia CESNET na optickej linke medzi Brnom a Viedňou s prenosovou kapacitou 10 Gb/s. Konkrétne išlo o linku spájajúcu sieť CESNET2 s Rakúskou sieťou Aconet. Graf zmien dátového toku na uvedenej linke v priebehu týždňa je zobrazený na obrázku 4.1. Zo zobrazeného grafu je možné vidieť, že linkou aj v najväčšej špičke tečie necelých 5 Gb/s a jej vyťaženie výrazne kolíše podľa dennej doby. Rozdiel vyťaženia je pozorovateľný aj medzi pracovnými dňami a víkendom.



Obr. 4.1: Vývoj dátového toku na linke CESNET2 – Aconet v priebehu týždňa

Akcelerácia sieťových aplikácií je najdôležitejšia počas najväčšieho vyťaženia linky, preto boli všetky testy realizované práve v jeho očakávanej dobe. Konkrétne boli pri jednotlivých meraniach sledované dáta počas 16 hodinového intervalu linky CESNET2 – Aconet vždy v pracovný deň v čase od 8:00 do 24:00. Samotné meranie bolo realizované využitím programu `flowmonexp` (FlowMon Exporter) firmy INVEA-TECH, ktorý je súčasťou softvérového vybavenia FlowMon sondy spomenutej v sekcii 2.6. Hlavnou výhodou uvedeného exportéra je jeho flexibilná architektúra umožňujúca použitie vlastných pluginov zabezpečujúcich vstup, spracovanie, filtrovanie a export dát. Na základe dát od vstupného pluginy exportér vytvára záznamy o tokoch a umožňuje ich ďalšie spracovanie procesnými pluginmi. Nakoniec je vo vhodnom čase možné záznamy o tokoch exportovať exportným pluginom. Pre potreby analýzy vlastností linky bola v rámci práce vytvorená zbierka špecifických procesných pluginov pre FlowMon Exporter, ktoré rozširovali informácie sledované

o jednotlivých sieťových tokoch o nové položky podľa potrieb konkrétneho merania. Ich výstupy boli potom spracované do podoby uvedenej v nasledujúcom texte pomocou súboru vlastných funkcií v programe `matlab`.

4.1 Základné vlastnosti

Základný pohľad na charakter dátového toku sieťovej linky je možné získať sledovaním rôznych štatistických informácií o ňom. Sledovať je možné priemerné objemové a časové parametre nosičov dát akými sú pakety alebo toky. Zaujímavý je tiež základný rozbor zastúpenia rôznych typov prebiehajúcich komunikácií, ktoré sú definovateľné použitým protokolom transportnej vrstvy a službou. Protokol je implicitne určený políčkom IP hlavičky paketov (Protocol alebo Next Header). Službu je zas možné odhadnúť na základe použitých čísel portov v UDP resp. TCP hlavičke. Rozlíšiteľné sú tak služby so známymi alebo registrovanými číslami portov (0 až 1023 a 1024 až 49 151).

V rámci realizovaného merania boli vzájomne rozlišované protokoly [24]: TCP (6), UDP (17) a ICMP (1) a podľa čísel portov služby [25]: HTTP (80), HTTPS (443), DNS (53), SMTP (25), SSH (22) a SIP (5060). Pre každú skupinu oboch rozlíšení sa sledovalo jej percentuálne zastúpenie z pohľadu tokov, paketov a bajtov na celkovom linkou prenesenom objeme. Sledované boli aj štatistické parametre ich paketov a tokov, konkrétne: priemerná veľkosť paketov, priemerná veľkosť tokov (v počte paketov) a priemerná doba trvania tokov. Namerané výsledky sú uvedené v tabuľke 4.1 pre protokoly resp. v tabuľke 4.2 pre služby.

	Tokov [%]	**Paketov [%]**	**Dát [%]**	**Tok [paketov]**	**Tok [s]**	**Paket [B]**
TCP	63.68	73.17	73.47	34.0	7.070	896.0
UDP	34.20	26.65	26.49	23.0	4.919	886.8
ICMP	1.91	0.13	0.01	1.9	3.206	91.3
ostatné	0.21	0.06	0.03	8.7	5.588	376.3
všetko	100.00	100.00	100.00	29.6	6.257	892.2

Tabuľka 4.1: Základné vlastnosti sieťových dát – protokoly

	Tokov [%]	**Paketov [%]**	**Dát [%]**	**Tok [paketov]**	**Tok [s]**	**Paket [B]**
HTTP	25.45	54.36	58.68	63.1	7.167	963.2
HTTPS	14.28	6.92	4.75	14.3	8.493	611.7
DNS	18.89	0.72	0.17	1.1	0.179	207.2
SMTP	0.38	0.22	0.14	17.2	2.934	573.8
SSH	0.04	0.01	0.00	11.6	17.433	233.0
SIP	0.00	0.00	0.00	4.9	24.701	420.9
ostatné	40.96	37.76	36.26	27.3	7.735	856.7
všetko	100.00	100.00	100.00	29.6	6.257	892.2

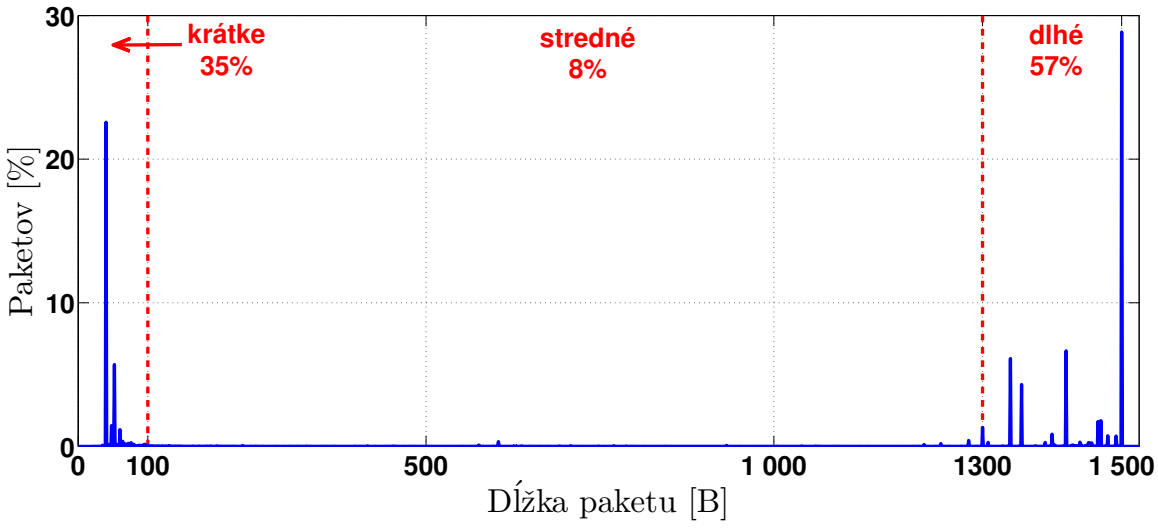
Tabuľka 4.2: Základné vlastnosti sieťových dát – služby

V poslednom riadku tabuliek je možné vidieť celkové priemerné vlastnosti nosičov dát v sieti. O sieťovom toku je tak možné štatisticky povedať, že sa skladá z paketov priemernej veľkosti 892 bajtov, ktoré sú združované v priemere po 30 do sieťových tokov trvajúcich priemerne niečo cez 6 sekúnd. Toto tvrdenie je možné ďalej spresňovať pre jednotlivé služby a protokoly ako je vidno v ostatných riadkoch tabuliek.

Z protokolov prevláda v sieťach jednoznačne TCP s približne $\frac{2}{3}$ zastúpením v počte tokov, paketov aj bajtov. Zvyšnú tretinu predstavuje prevažne UDP a čiastočne ICMP, ostatné protokoly predstavujú nevýznamné percento. Zaujímavým faktom pozorovateľným v tabuľke 4.1 sú veľmi ľahké ICMP toky. Oproti celkovému priemeru obsahujú výrazne menej paketov, ktoré sú navyše menšie oproti priemeru. To má za následok podstatne väčšie percentuálne zastúpenie ICMP v počte tokov oproti počtu paketov a bajtov. Ďalšou pozorovateľnou vlastnosťou je, že TCP toky sú obecne ťažšie ako UDP toky.

Zo služieb výrazne prevláda používanie HTTP s nadpolovičnou väčšinou paketov a bajtov, a tiež asi štvrtinovým zastúpením v počte tokov. Toky aj pakety patriace HTTP sú zároveň obecne výrazne ťažšie ako tie patriace ostatným službám. Veľký podiel sieťového toku pripadá aj službe HTTPS, ktorá predstavuje šifrovaný variant HTTP. Veľkosť jej tokov je však podstatne menšia. Ešte výraznejší nepomer počtu pozorovaných tokov k objemu prenesených paketov a bajtov ako pre protokol ICMP je pozorovaný u služby DNS. DNS toky tvoriace asi pätinu všetkých sieťových tokov, sú extrémne ľahké a v priemere obsahujú len jediný paket malej veľkosti.

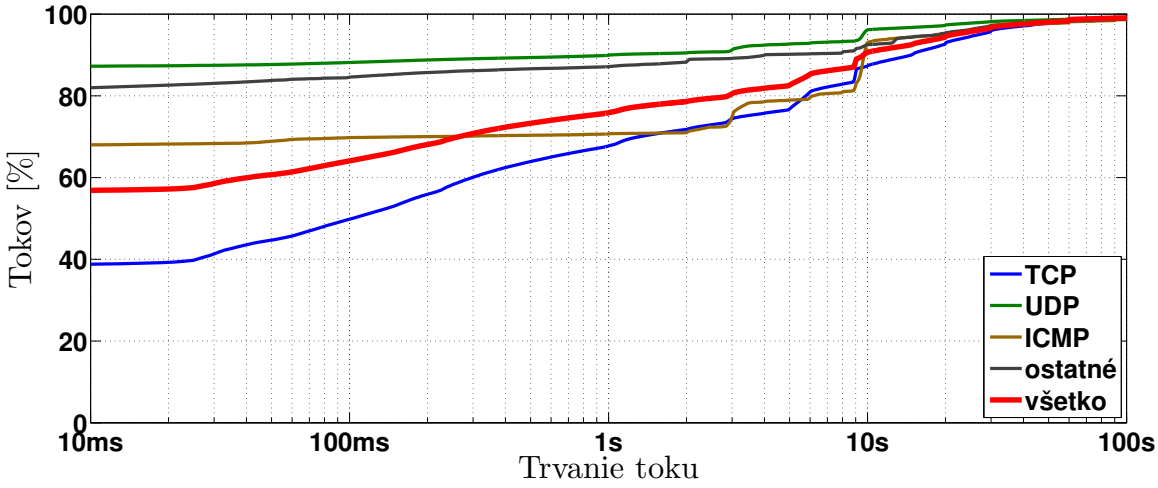
Okrem uvedeného rozboru základných vlastností protokolov je zaujímavý aj bližší pohľad na používanie rôznych dĺžok paketov v sieti. Ethernet umožňuje použitie rámcov nesúcich pakety ľubovoľných dĺžok z rozsahu 42 až 1500 B. Zmerané percentuálne zastúpenie ich jednotlivého využitia je zobrazené v grafe na obrázku 4.2. Z grafu je na prvý pohľad jasné, že výrazne prevládajú veľmi krátke (35 %) a veľmi dlhé pakety (57 %). Najdominantnejšie je pritom zastúpenie oboch extrémnych dĺžok. Na druhej strane pakety stredných dĺžok, kam spadá aj priemerná veľkosť paketu, sú reálne používané zriedka a predstavujú len 8 % z celkového počtu.



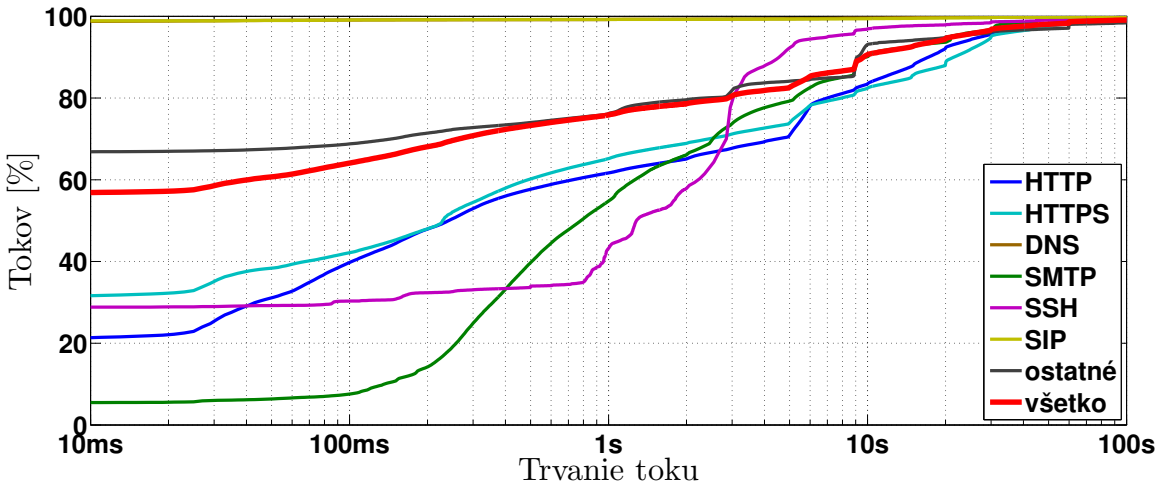
Obr. 4.2: Rozloženie hustoty výskytu paketov podľa veľkosti

4.2 Časové vlastnosti tokov

V predošlej sekcii už bolo zmerané trvanie priemerného sieťového toku na 10 Gb/s linke na niečo viac ako 6 sekúnd. Podrobnejší náhľad na doby trvania tokov pre jednotlivé protokoly resp. služby ponúkajú grafy na obrázku 4.3 resp. 4.4. Ide o grafy zachycujúce na základe meranie vytvorené distribučné funkcie časového trvania tokov. Na ich základe je všeobecne (červená závislosť) možné povedať, že až $\frac{2}{3}$ zo všetkých sieťových tokov nemá dlhšie trvanie ako 100 ms a len asi desatina ich prekročí 10 s.



Obr. 4.3: Distribučné funkcie trvania tokov – protokoly



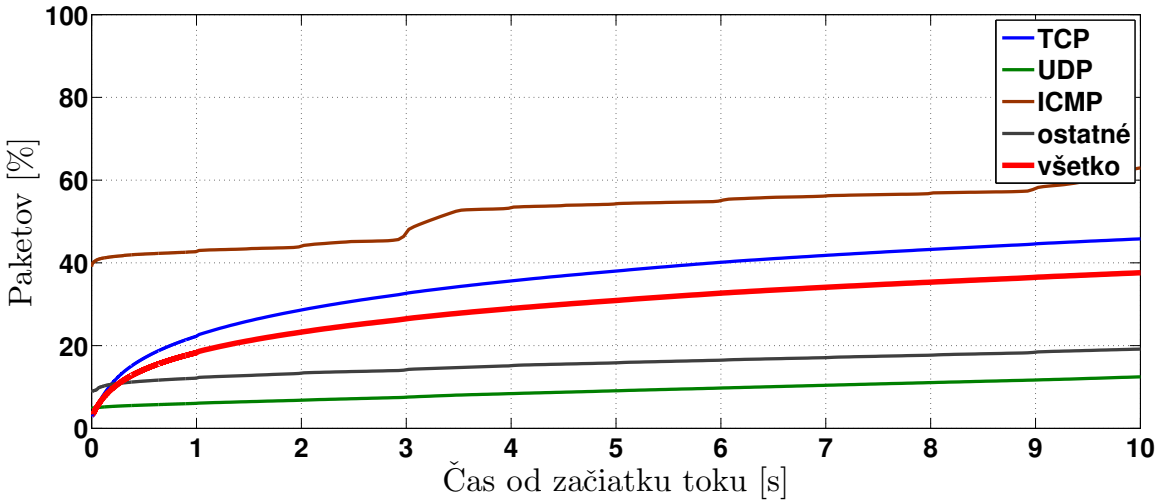
Obr. 4.4: Distribučné funkcie trvania tokov – služby

V grafe protokolov (4.3) je ďalej možné vidieť, že väčšina UDP tokov má veľmi krátke trvanie. Na druhej strane TCP toky sú oproti ostatným rádovo dlhšie, pretože len polovica z nich trvá kratšie ako 100 ms. Zaujímavým pozorovaním je aj zarovnanie trvania veľkého počtu ICMP tokov na násobky 3 sekúnd. Viditeľné to je vo výraznejších skokoch distribučnej funkcie ICMP v čase 3, 6 a hlavne 9 s. V grafe služieb (4.4) je zase možné pozorovať, že DNS a SIP toky sú skoro všetky veľmi krátkeho trvania. Naopak SMTP väčšinou pracuje s dlhšie trvajúcimi tokmi a veľmi krátkych má len relatívne malý počet. Distribučné funkcie

trvaní HTTP a HTTPS tokov sú si vcelku podobné a tieto služby patria tiež k tým s väčším počtom dlhšie trvajúcich tokov. Nakoniec služba SSH je zvláštna tým, že aj keď istá časť (asi 30 %) jej tokov skončí do 10 ms tak majoritný počet z nich trvá rádovo jednotky sekúnd.

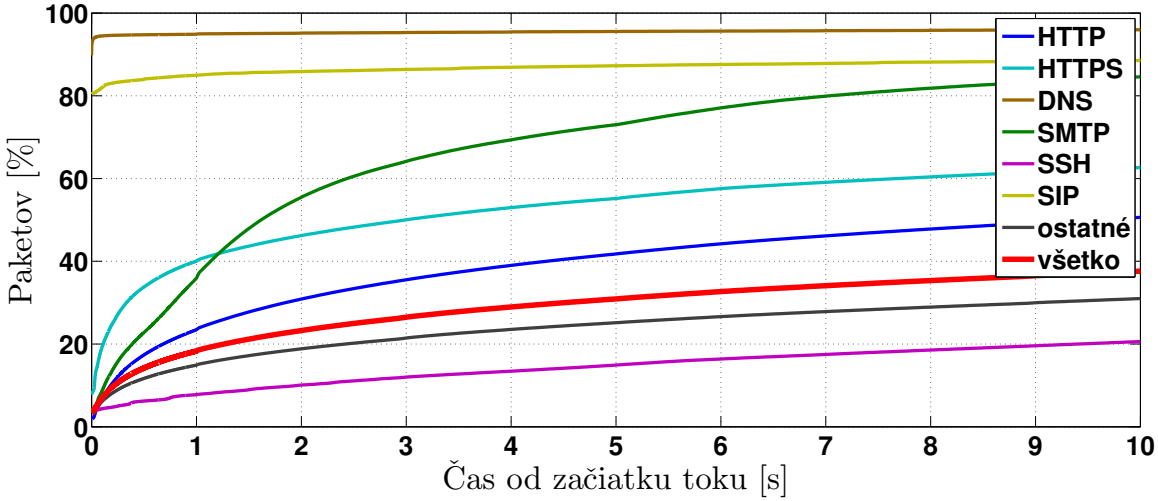
Popísané grafy distribučných funkcií trvania tokov síce poskytujú bližší náhľad na časové vlastnosti tokov jednotlivých protokolov a služieb, ale nevypovedajú nič o časovom rozložení príchodov paketov v rámci tokov. Z nameraných hodnôt tak nevieme určiť či toky prenášajú pakety viac menej rovnomerne počas celého svojho trvania alebo prenesú väčšinu z nich na začiatku/konci. Taktiež nie je zohľadnené, ktoré toky sú ako ťažké. Lepší pohľad na časovanie paketov v rámci tokov môže priniesť meranie relatívnych časov príchodu paketov vzhľadom na začiatok toku. Teda prvý paket každého toku má relatívny čas príchodu rovný nule a každý ďalší paket daného toku má relatívny čas príchodu rovný rozdielu absolútnych časov jeho príchodu a príchodu prvého paketu.

Na základe vyššie popísaných meraní relatívnych časov príchodu paketov v tokoch boli vytvorené grafy distribučných funkcií na obrázkoch 4.5 (protokoly) a 4.6 (služby). Zachytené funkcie majú výrazne odlišný charakter priebehu ako distribučné funkcie trvaní tokov. Z grafov je viditeľné, že len malá časť všetkých paketov je prenesená v tokoch hneď po ich začiatku. Napríklad za prvú sekundu tokov je priemerne prenesená len pätina paketov. Toto zistenie vedie k záveru, že krátko trvajúce toky prenášajú väčšinou len po veľmi málo paketov. Sila uvedeného dôsledku je ešte znásobená faktom, že počet krátko trvajúcich tokov predstavuje veľkú väčšinu z celkového počtu.



Obr. 4.5: Distribučné funkcie časov výskytu paketov v tokoch – protokoly

V grafe protokolov (4.5) je ďalej možné vidieť, že napríklad UDP toky prenášajú pakety relatívne rovnomerne počas celého svojho trvania (lineárne rastúci graf distribučnej funkcie). Naproti tomu ICMP toky prenesú veľkú časť celkového počtu paketov rýchlo po začiatku. Spôsobené to je malým počtom dlhých tokov. V príchodoch paketov ICMP tokov je aj teraz možné pozorovať ich zarovnanie na trojsekundové intervaly. V grafe služieb (4.6) je možné pozorovať, že veľmi výrazná časť z celkového počtu paketov DNS a SIP tokov je prenesená rýchlo po ich začiatku. Aj v tomto prípade je to dôsledok malého počtu dlhých DNS a SIP tokov. Veľká časť paketov je relatívne rýchlo po začatí tokov prenesená aj v SMTP. Nakoniec v SSH tokoch sa pakety vyskytujú v zásade rovnomerne počas celého ich trvania.



Obr. 4.6: Distribučné funkcie časov výskytu paketov v tokoch – služby

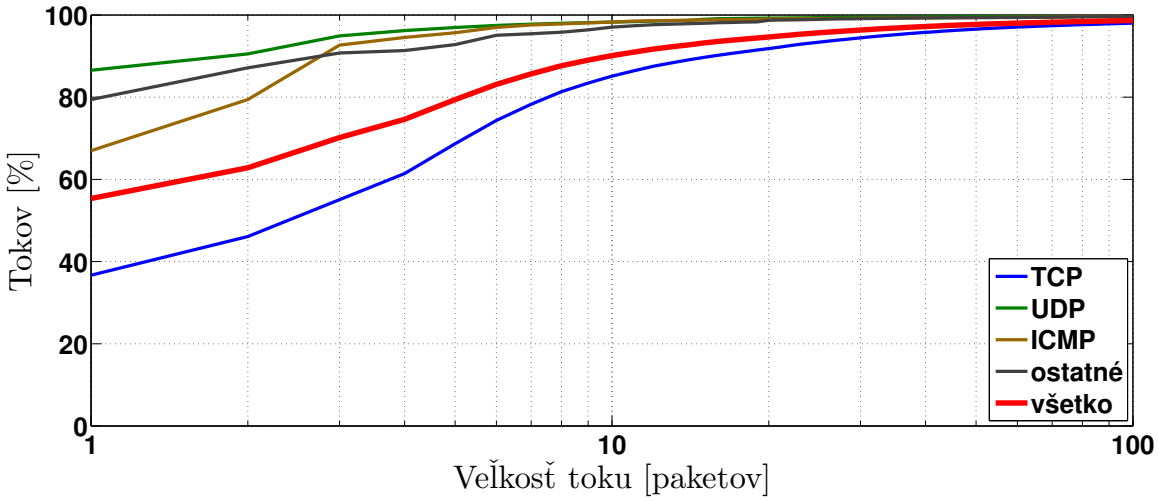
V sekcii obsiahnuté grafy 4.4 a 4.6 zobrazujú distribučné funkcie jednotlivých sieťových služieb nezávisle na smere ich tokov. Teda napríklad do distribučnej funkcie HTTP služby sú započítané pakety smerujúce na server (cieľový port 80) aj prichádzajúce od neho (zdrojový port 80). Smerovo závislé varianty uvedených grafov je možné nájsť v prílohe A na obrázkoch A.1 až A.4. Priebehy funkcií sú v smerových variantoch grafov trochu odlišné, ale väčšinou v nich nie je vidieť výraznejšie zmeny od obojsmernej verzie. Výnimkou sú len služby SIP a SSH v smere od serveru (zdrojový port je SIP resp. SSH).

4.3 Objemové vlastnosti tokov

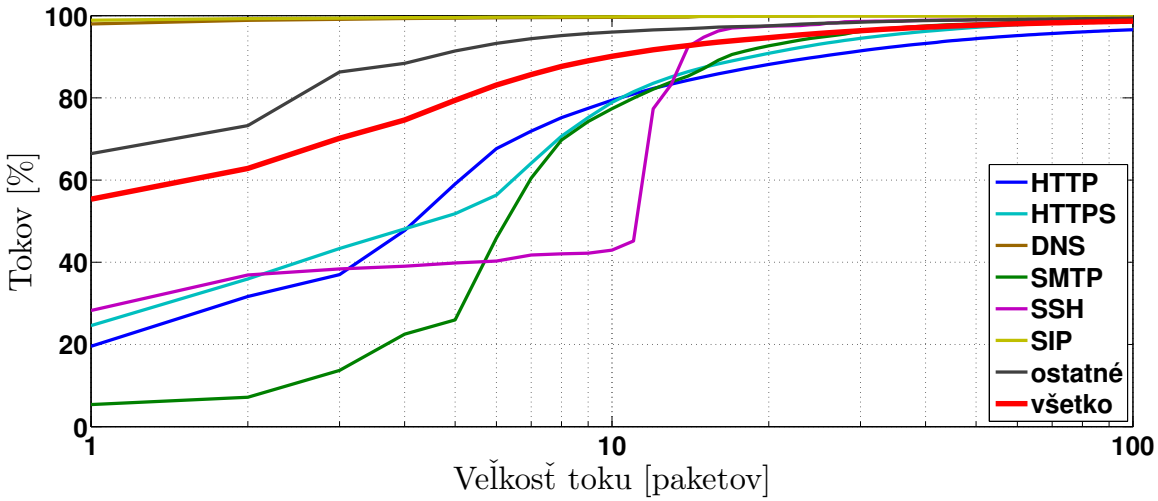
V sekcii 4.1 už bola zmeraná veľkosť priemerného sieťového toku na 10 Gb/s linke na približne 30 paketov. Podrobnejší náhľad na veľkosti tokov pre jednotlivé protokoly resp. služby ponúkajú grafy na obrázku 4.7 resp. 4.8. Ide o grafy zachycujúce na základe meranie vytvorené distribučné funkcie veľkosti tokov v počte paketov. Na ich základe je všeobecne (červená závislosť) možné povedať, že len desatina zo všetkých sieťových tokov obsahuje viac ako 10 paketov.

V grafe protokolov (4.7) je ďalej možné vidieť, že väčšina UDP a ICMP tokov nesie veľmi málo paketov. Na druhej strane TCP toky sú oproti ostatným rádovo ťažšie. V grafe služieb (4.8) je zase možné pozorovať, že skoro všetky DNS a SIP toky sú tvorené len jediným paketom. Na druhej strane SMTP väčšinou pracuje s aspoň 5 paketovými tokmi a jedno paketových má len relatívne malý počet. Distribučné funkcie veľkostí HTTP a HTTPS tokov sú si vcelku podobné a obe tieto služby patria k tým s väčším počtom ťažších tokov. Nakoniec služba SSH je zvláštna tým, že aj keď istá časť (asi 30 %) jej tokov obsahuje len 1 paket tak majoritný počet z nich obsahuje 10 až 20 paketov.

Popísané grafy distribučných funkcií veľkosti tokov síce poskytujú bližší náhľad na objemové vlastnosti tokov jednotlivých protokolov a služieb, ale nie je z nich jasne viditeľný podiel tokov jednotlivých veľkostí na celkovom počte prenesených paketov. O vysokorýchlostnom sieťovom toku je známe, že má heavy-tailed charakter rozloženia veľkostí tokov. Vo výsledku to znamená, že aj malé percento najťažších tokov obsahuje vysoké percento všetkých paketov. Heavy-tailed charakter rozloženia počtu paketov v tokoch odvodený z na-



Obr. 4.7: Distribučné funkcie veľkosti tokov – protokoly

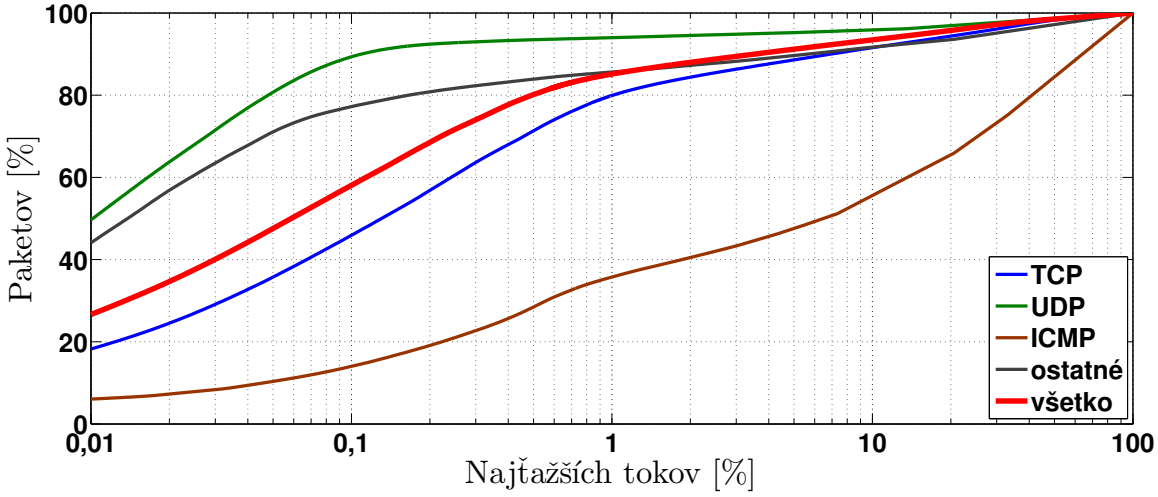


Obr. 4.8: Distribučné funkcie veľkosti tokov – služby

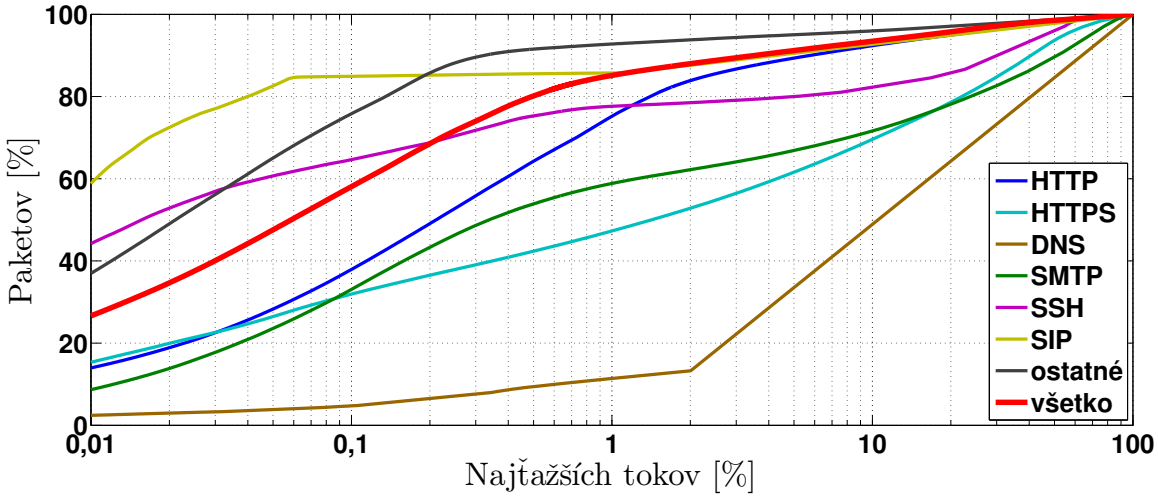
meraných dát je zachytený v grafoch na obrázkoch 4.5 (protokoly) a 4.6 (služby). Grafy zobrazujú distribučné funkcie vytvorené integrovaním funkcií podielu paketov obsiahnutých v jednotlivých tokoch zoradených zostupne podľa veľkosti. Z grafov je viditeľné, že aj malé percento najväčších tokov obsahuje veľkú časť paketov. Napríklad promile najťažších tokov obsahuje približne 60 % všetkých paketov a percento najťažších tokov až vyše 85 % paketov.

V grafe protokolov (4.5) je ďalej možné vidieť, že napríklad UDP toky majú oproti ostatným viac preťažené najťažšie toky. Naproti tomu ICMP služba prenesie veľkú časť celkového počtu paketov v relatívne ľahkých tokoch. Spôsobené to je veľkým počtom tokov s jediným paketom. V grafe služieb (4.6) je možné pozorovať, že veľmi výrazná časť (asi 95 %) z celkového počtu paketov DNS je prenesená v najľahších tokoch. Aj v tomto prípade je to dôsledok veľkého počtu DNS tokov s jediným paketom. Naproti tomu SIP a v istej miere aj SSH majú extrémne preťažených niekoľko najťažších tokov.

V sekcii obsiahnuté grafy 4.8 a 4.10 zobrazujú distribučné funkcie jednotlivých sieťových služieb nezávisle na smere ich tokov. Teda napríklad do distribučnej funkcie HTTP služby sú



Obr. 4.9: Heavy-tailed roloženie počtu paketov v tokoch – protokoly



Obr. 4.10: Heavy-tailed rozloženie počtu paketov v tokoch – služby

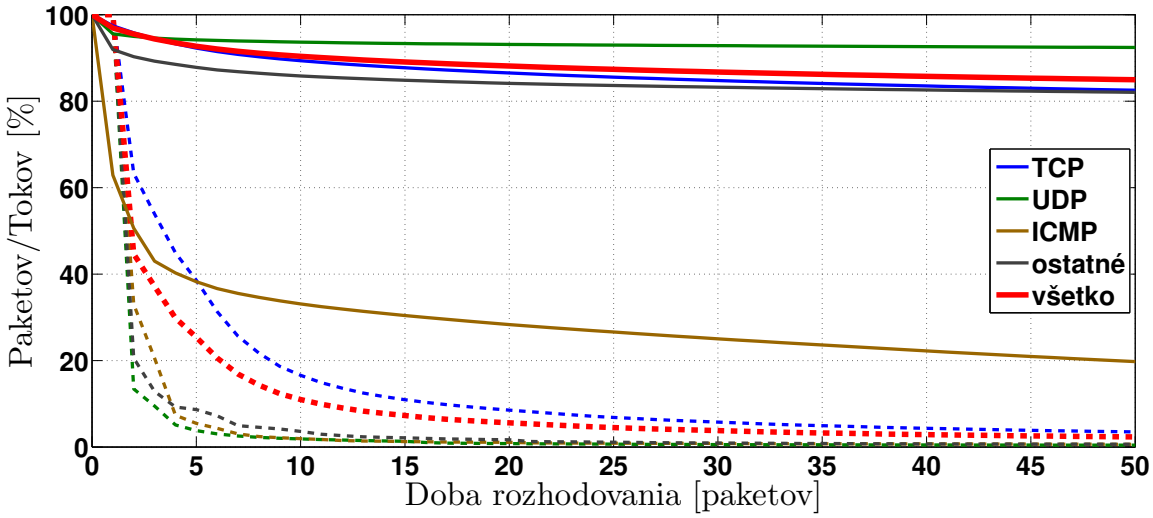
započítané pakety smerujúce na server (cieľový port 80) aj prichádzajúce od neho (zdrojový port 80). Smerovo závislé varianty uvedených grafov je možné nájsť v prílohe A na obrázkoch A.5 až A.8. Priebehy funkcií sú v smerových variantoch grafov trochu odlišné, ale väčšinou v nich nie je vidieť výraznejšie zmeny od obojsmernej verzie. Výnimkou sú len služby SIP a SSH v smere od serveru (zdrojový port je SIP resp. SSH).

4.4 Detekovanie ťažkých tokov

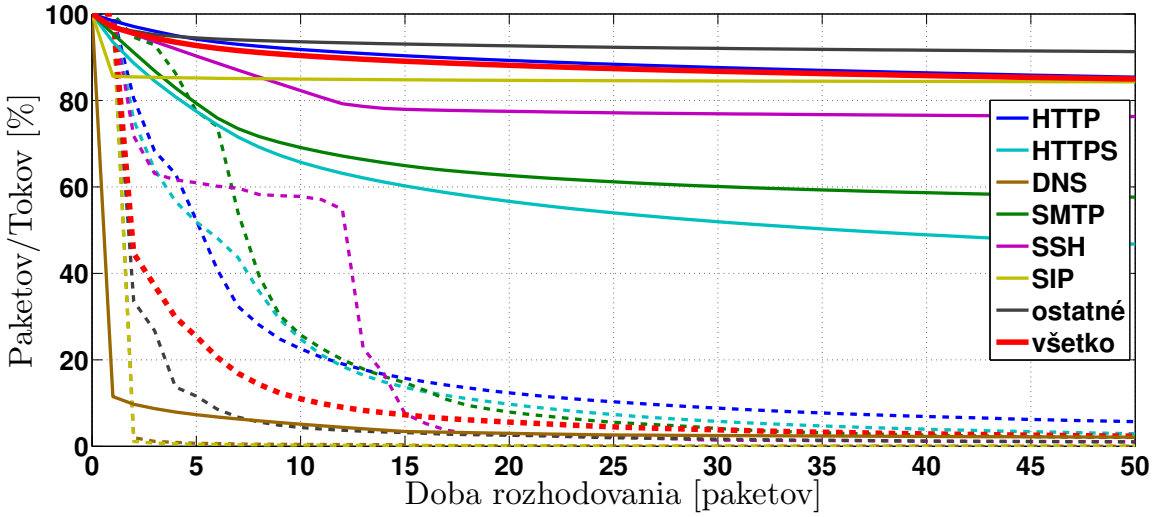
V predošlej sekcii bol ukázaný heavy-tailed charakter rozloženia počtu paketov v sieťových tokoch. Výberom a spracovaním len niekoľkých málo z najväčších tokov je vďaka tomu možné pokryť veľkú časť dát tečúcich linkou. Problémom však zostáva vhodný spôsob efektívnej predpovede, či konkrétny tok bude z pomedzi aktuálne najťažších na linke alebo nie. Inak povedané, ide o schopnosť rozpoznávať potenciálne ťažké toky len na základe pozorovania vlastností ich prvých niekoľkých paketov. Matematicky je úlohu možné formulovať

ako výber množiny maximálne n sieťových tokov zo všetkých aktuálne prebiehajúcich len na základe pozorovateľných vlastností istého zvoleného počtu k prvých paketov tokov. Cieľom pritom je maximalizovať hodnotu $G = \sum_{i=1}^{n}(\max(0, s_i - k))$, kde s_i označuje počet paketov i-teho vybraného toku a samotné G predstavuje počet výberom pokrytých paketov (všetky po k-tom vo vybraných tokov).

Najjednoduchší spôsob realizovania popísanej predikcie veľkých tokov je založený na naivnom pravidle, že každý tok, ktorý má aspoň zvolených k paketov je potenciálne ťažký. Teda jedinou pozorovanou vlastnosťou prvých paketov toku je ich samotná existencia. Výhodou takéhoto rozhodovania je jeho jednoduchosť nevyžadujúca analýzu paketov a ukladanie zložitej stavovej informácie o tokoch. Otázna je ale presnosť jej schopnosti predikcie ťažkých tokov.

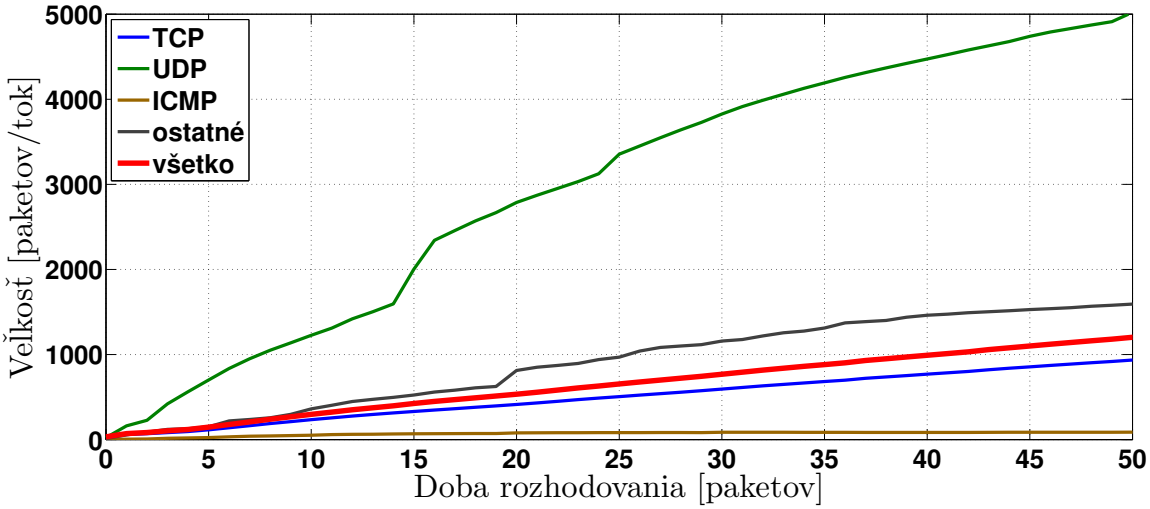


Obr. 4.11: Výsledky detekcie ťažkých tokov naivnou metódou – protokoly



Obr. 4.12: Výsledky detekcie ťažkých tokov naivnou metódou – služby

Aplikovaním popísanej naivnej metódy detekcie ťažkých tokov na odmerané parametre sieťového toku na vysokorýchlostnej linke pre rôzne hodnoty k boli vytvorené grafy na obrázkoch 4.11 a 4.12. Na os x je vynesený počet paketov použitých na rozhodovanie (spomínané k). Na osi y je percentuálny podiel vybraného počtu tokov z celkového počtu (čiarkované čiary) alebo podiel hodnoty G z naivnou metódou vybraných tokov a celkového počtu paketov (plné čiary). Spojením oboch vynesených údajov pre konkrétny protokol alebo službu je možné získať predstavu o efektivite požitej metódy. Výrazná redukcia počtu tokov pri len pozvoľnej redukcii pokrytého podielu paketov ukazuje na relatívne kvalitné a použiteľné výsledky. Grafy napríklad ukazujú, že pre $k = 20$ je zo všetkých tokov za ťažké označených len asi 5 %, pričom je tým pokrytých až 85 % paketov. Z grafov je ďalej vidno, že naivná metóda nefunguje veľmi dobre pre ICMP protokol a DNS službu, kde s redukciou počtu tokov je výrazne redukovaný aj počet pokrytých paketov. Vcelku výrazná redukcia pokrytia paketov pri redukcii počtu vybraných tokov je pozorovateľná aj pre služby SMPT a HTTPS.

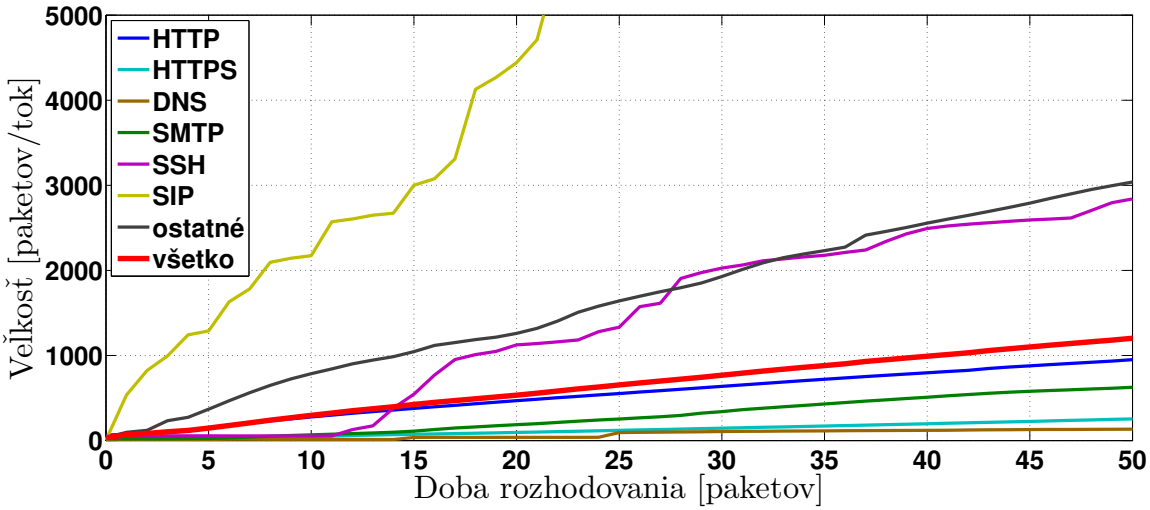


Obr. 4.13: Priemerný počet paketov na tok pri naivnej metóde – protokoly

Iný náhľad na efektivitu použitej metódy výberu ťažkých tokov je možné získať aj na základe pomeru počtu pokrytých paketov a počtu vybraných tokov. Uvedený pomer udáva priemerný počet paketov pokrytých jedným zvoleným tokom. Konkrétne hodnoty pri použití naivnej metódy sú vynesené do grafov na obrázkoch 4.13 a 4.14. Diametrálny rozdiel v grafoch zachytených hodnôt a nameranej priemernej veľkosti všeobecného toku (30 paketov) jednoznačne dokazuje, že sú vyberané rádovo najväčšie z tokov resp. sú eliminované rádovo najmenšie. Grafy tiež ukazujú rozdiel efektivity výberu tokov pre rôzne protokoly a služby. UDP, SSH a SIP toky sú najvhodnejšie na výber, DNS a ICMP zase najmenej vhodné.

V sekcii zobrazené grafy 4.12 a 4.14 upresňujúce skúmané závislosti pre jednotlivé sieťové služby sú smerovo nezávislé. Ich smerovo závislé varianty je možné nájsť v prílohe A na obrázkoch A.9 až A.12.

Celkovú efektivitu použitia naivnej metódy je možné ďalej zlepšovať rôznym upresňovaním výberu. Jednou z ponúkajúcich sa možností je napríklad použitie rôznych hodnôt rozhodovacieho počtu paketov k pre rôzne protokoly alebo služby. Všeobecne je hodnotu k možné dynamicky ovplyvňovať za behu pre jednotlivé toky na základe pozorovaných vlastností už pre ne prijatých paketov. V dizertačnej práci [26] je ukázané, že klasifikácia tokov



Obr. 4.14: Priemerný počet paketov na tok pri naivnej metóde – služby

do tried podľa veľkosti je veľmi presne možná na základe parametrov ich paketov ako sú najmä: veľkosť, protokol, čísla portov (služba) alebo TCP príznaky (Code Bits). Preto je očakávané, že popísanú naivnú metódu by bolo možné výrazne spresniť práve vhodným rozhodovaním na základe uvedených vlastností.

4.5 Zhrnutie z pohľadu SDM

Na základe výsledkov meraní uvedených v predošlých častiach kapitoly je možné zhodnotiť dopad potenciálnych slabých miest prezentovaných v sekcii 3.3 na fungovanie navrhovaného akceleračného modelu SDM. Konkrétne je pre slabé miesta SDM možné vyvodiť nasledujúce závery:

Dlhé trvanie spätnej väzby. Časové trvanie spätnej väzby SDM je možné rádovo očakávať v desiatkach až stovkách milisekúnd. Výsledky uvedené v sekcii 4.2 síce ukazujú, že väčšina tokov trvá príliš krátko anto, aby stihli byť včas vyhodnotené – $\frac{2}{3}$ tokov skončia do 100 ms. Na druhej strane však vidno, že veľká väčšina zo všetkých paketov je prenášaná dlhšie trvajúcimi tokmi a v neskoršom čase od ich začiatku – počas prvých 100 ms je prenesená asi len desatina paketov. Uvedené výsledky vedú k očakávaniu malého dopadu oneskorenia rozhodovania pre systém SDM. Výnimkou sú toky protokolu ICMP a služieb DNS a SIP.

Obmedzená kapacita firmvéru. V sekcii 4.3 je ukázaný heavy-tailed charakter rozloženia veľkostí sieťových tokov. Sekcia 4.4 zároveň demonštruje, že aj s použitím veľmi jednoduchej metódy je dosiahnuté efektívne rozpoznávanie najťažších zo sieťových tokov. Vo výsledku je preto aj použitím nízkeho počtu pravidiel pre toky možné pokryť majoritnú časť paketov – výberom napríklad len 5 % tokov je pokrytých 85 % paketov. Výnimkou sú toky protokolu ICMP a služby DNS.

Nedostatočná redukcia dát. Veľkosť unifikovanej hlavičky paketu alebo záznamu o toku je v rádoch desiatok bajtov. Sekcia 4.1 ukazuje výraznú prevahu použitia veľmi dlhých paketov vedúceho k priemernej veľkosti paketu skoro až 900 B. Dosiahnuteľná redukcia

veľkosti paketov a tak aj dátového toku do softvéru je preto výrazná. Výnimkou je protokol ICMP, ktorého pakety sú prevažne veľmi krátke.

Vysoká jemnosť kontroly. Dosiahnuteľná efektivita zavádzania jednotlivých pravidiel o tokoch do hardvéru je popísaná v sekcii 4.4. Ukázané je, že vhodným výberom tokov je možné dosiahnuť spracovanie priemerne stoviek až tisícov paketov na každé zavedené pravidlo. Výnimkami sú v tomto prípade toky protokolu ICMP a služieb DNS a HTTPS.

Z vyššie uvedených záverov analýzy vlastností sieťových dát na vysokorýchlostnej linke vyplýva, že potenciálne slabé miesta návrhu SDM nemajú veľký dopad na dosiahnuteľnú efektivitu akcelerovania spracovania väčšiny paketov v tokoch. Problém predstavuje v zásade len služba DNS a protokol ICMP. Oba sa vyznačujú veľmi ľahkými tokmi a hlavne obrovským podielom tokov s jediným paketom. Akékoľvek rozhodovanie o predspracovaní jednopaketových tokov až po ich začiatku (spracovaní prvého a jediného paketu) je nemožné. Jediným riešením tohto problému je pridanie podpory dopredu definovaných statických pravidiel riadiacich predspracovanie istých skupín tokov. Nové (neznáme) toky by tak mohli byť rôzne predspracované už na základe statických pravidiel. Za behu zavádzané softvérové pravidlá pre jednotlivé toky by potom mali vyššiu prioritu a ďalej by upresňovali predspracovanie konkrétnych tokov. Príkladom môže byť klasické monitorovanie na bázy tokov, ktorému pre všetky pakety stačia len unifikované hlavičky (statické pravidlo), a ktoré môže spracovanie zvolených potenciálne ťažkých tokov ešte viac urýchliť presunom ich predspracovania do hardvérovej cache tokov (dynamické pravidlá). Zavedením statických pravidiel pre skupiny tokov je tak možné pre niektoré aplikácie dosiahnuť lepšiu akceleráciu spracovania sieťových dát.

Kapitola 5

Podrobný implementačný návrh

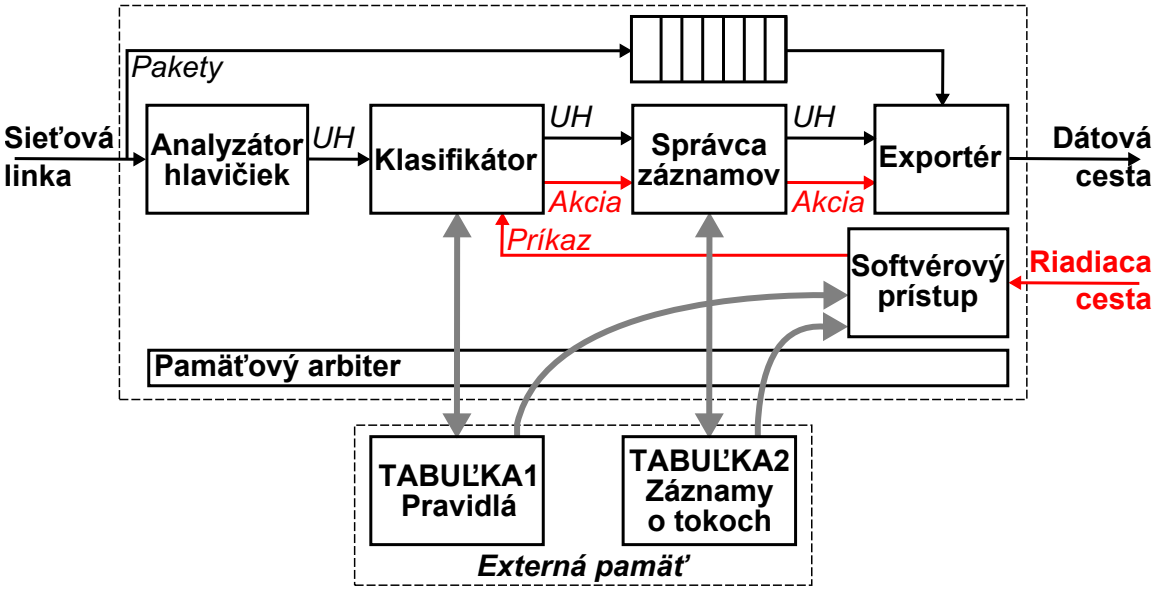
Kapitola popisuje podrobný implementačné orientovaný návrh softvérovo riadeného akceleračného systému vytvoreného na základe konceptu Software Defined Monitoring predstaveného v kapitole 3. Koncept je ďalej upravený podľa vyhodnotenia výsledkov analýzy vlastností vysokorýchlostných sieťových tokov uvedenej v kapitole 4. Popis je rozdelený na firmvérovú a softvérovú časť systému. Podrobný návrh firmvérovej časti je zameraný na vlastnosti potrebných funkčných jednotiek systému, ich rozhranie a spôsob zapojenia do celkovej architektúry. Detaily vhodnej realizácie funkcionality jednotlivých komponentov nie sú bližšie rozoberané. Návrh softvérovej časti systému je sústredený okolo realizácie SDM radiča (kontrolóra) a komunikácie s ním zo strany užívateľských aplikácií.

5.1 Firmvérová časť

Hlavnou požadovanou funkcionalitou firmvérovej architektúry SDM, fungujúcej v čipe FPGA na hardvérovej sieťovej karte, je softvérovo riadené predspracovanie dátového toku z vysokorýchlostnej siete. Realizovateľné je zreťazenou procesnou linkou rozdelenou na štyri základné jednotky: analýzu hlavičiek rámcov, klasifikáciu a výber akcie podľa pravidiel, samotné predspracovanie a export do softvéru v zvolenom formáte. Významným parametrom architektúry je softvérové riadenie predspracovania. Realizované je v rámci špeciálne na to určenej jednotky, ktorá je schopná ďalej konfigurovať jednotlivé prvky architektúry. Ďalším dôležitým parametrom firmvéru je dostatočná kapacita pre pravidlá na klasifikáciu rámcov a pre záznamy o tokoch, ktoré sú vytvárané ako jedna z možností predspracovania. Pre tento účel je potrebné ukladanie riešiť v externej pamäti. Riadenie a sprostredkovanie prístupu z jednotlivých jednotiek v FPGA do tejto pamäte bude zabezpečovať pamäťový arbiter.

Zapojenie popísaných firmvérových jednotiek do celkovej architektúry znázorňuje schéma na obrázku 5.1. V schéme sú dátové cesty znázornené čiernymi, riadiace informácie červenými a pamäťové prístupy sivými šípkami. V hornej časti schémy je možné vidieť procesnú linku spracúvajúcu vstupný sieťový tok a generujúcu výstupný dátový tok do softvéru. Dáta samotných rámcov nie sú touto linkou šírené, ale sú uchovávané vo FIFO pamäti paralelne k nej. Linka pracuje len so zaujímavými dátami extrahovanými z hlavičiek rámcov (UH). V dolnej časti schémy je znázornená externá pamäť rozdelená na 2 časti – tabuľku s pravidlami a tabuľku so záznamami o tokoch. Ako je možné vidieť, všetky prístupy k obom tabuľkám externej pamäti sú sprostredkovávané jednotkou pamäťového arbitra. Podobne je

viditeľné, že všetky konfiguračné informácie prichádzajúce zo softvéru po riadiacej ceste sú spracúvané a sprostredkovávané jedinou na to určenou jednotkou.



Obr. 5.1: Celková schéma SDM firmvéru

Pred popisom presnejšej funkcionality samotných jednotiek je dôležité pristaviť sa pri spôsobe akým sú realizované dátové cesty architektúry. Jeho návrh sa môže na prvý pohľad javiť triviálny, avšak pre dosiahnutie vysokej priepustnosti potrebnej na spracovanie 100 Gb/s dátového toku sa stáva kľúčovým. Dôvodom je dosiahnuteľná pracovná frekvencia FPGA dizajnu, ktorá neprekračuje 400 MHz. Na dosiahnutie teoretickej priepustnosti aspoň 100 Gb/s je tak potrebné využiť veľmi širokú dátovú cestu (512 b pri 200 MHz alebo 1024 b pri 100 MHz). Keďže ale najkratší Ethernetový rámec má 64 B (512 b), aliasing a zarovnávanie môžu predstavovať veľký problém pre dosahovanú efektívnu priepustnosť, ktorá tak môže byť podstatne menšia ako teoretická. Napríklad prenos 65 B rámca by štandardne vyžadoval 2 dátové slová, kde z druhého by sa nevyužilo až 63 B. Efektivita využitia zbernice by tak nebola o moc vyššia ako polovičná. Výrazné zvýšenie efektivity prenosu rámcov po dátovej zbernici je možné dosiahnuť využitím dvoch navrhnutých techník:

Čiastočné zarovnanie začiatku. Prvý bajt rámcov sa môže vyskytnúť na ľubovoľnej pozícii zarovnanej na násobok 8 bajtov. To korešponduje so zarovnaním definovaným v 40 a 100 GbE štandarde. Pre široké dátové slová tak rámec môže začínať na viacerých miestach v slove ako len na jeho začiatku.

Zdieľanie dátových slov. V jednom dátovom slove môžu byť obsiahnuté posledné bajty istého rámca X a zároveň prvé bajty nasledujúceho rámca $X+1$. Rámce sa samozrejme nesmú v slove prekrývať a musí byť splnená vyššie definovaná podmienka čiastočného zarovnania začiatku rámcu $X + 1$.

Pri použití uvedených dvoch techník je možné dosiahnuť prenos ľubovoľných rámcov dlhších ako šírka slova s nevyužitím maximálne len 7 B zo slova. To vedie k vyše 90 % efektivite prenosu rámcov aj v najhoršom prípade.

Ako zobrazuje vyššie popísaná schéma 5.1, potrebnú funkcionalitu SDM firmvéru je možné rozdeliť do 6 základných funkčných jednotiek. Podrobnejší popis fungovania týchto jednotiek je uvedený v nasledujúcom texte.

Analyzátor hlavičiek. Jednotka rieši analýzu jednotlivých hlavičiek vstupných sieťových rámcov a extrakciu zaujímavých informácií z nich. Extrahovať je potrebné hlavne políčka jednoznačne identifikujúce sieťový tok. Analyzovať je preto potrebné hlavičky protokolov až po aplikačnú vrstvu. Pre nasadenie v dnešných sieťach je tak potrebné vedieť spracovať aspoň najrozšírenejšie používané hlavičky, konkrétne: Ethernetovú, VLAN, MPLS, IPv4, IPv6 (s rozširujúcimi hlavičkami), UDP a TCP. Výstupom analyzátora sú extrahované dáta jednotlivých rámcov poskladané do istého unifikovaného formátu.

Klasifikátor. Hlavnou úlohou je schopnosť priradiť akciu každému rámcu na základe z neho extrahovaného identifikátoru toku. Priradenie akcií rámcom je realizované na základe množiny nastaviteľných pravidiel tvaru: identifikátor toku a akcia (tabuľka 1 v externej pamäti). Správa pravidiel je zabezpečená cez vstupné kontrolné rozhranie podporujúce príkazy na pridávanie, odoberanie a editovanie pravidiel. Toto rozhranie podporuje aj príkazy určené pre správcu záznamov o tokoch, ktoré sú klasifikátorom v prípade potreby len predspracované a odoslané ďalej. V jednotke je potrebná podpora veľkého počtu pravidiel (rádovo tisíce až milióny), preto je na ich ukladanie využitá externá pamäť. Samotná klasifikácia rámcov a výber akcie pre ne prebieha v dvoch fázach. Najprv sú klasifikované využitím množiny relatívne malého počtu statických pravidiel nad skupinami tokov. Následne môže byť takto získaná akcia ešte spresnená klasifikáciou podľa množiny dynamických pravidiel pre konkrétne toky. Prvý krok klasifikácie je možné implementovať využitím malej asociačnej pamäte, druhý krok napríklad vhodnou formou haš tabuľky.

Správca záznamov. Jednotka sa stará o správu záznamov o tokoch uložených v tabuľke 2 externej pamäti. Hlavne o aktualizáciu ich hodnôt podľa vstupnej UH na základe k nej patriacej akcie. Akcia v sebe musí niesť adresu záznamu, s ktorým sa má pracovať a identifikáciu samotnej operácie, ktorá sa s ním má vykonať. Všeobecne je možné podporovať viacero typov operácií a dokonca viacero typov záznamov o tokoch. V základnej verzii postačí agregačná operácia využívaná u prostého monitorovania tokov, teda zvýšenie počítadiel paketov/bajtov, prepis koncovej/počiatočnej časovej značky a logický or TCP príznakov. Vykonanie úpravy záznamu danou operáciou je štandardne potrebné realizovať 2 krokmi: načítanie záznamu z pamäte a zápis upraveného záznamu späť. Okrem samotných funkčných operácií nad záznamami je potrebné podporovať aj operácie špeciálneho významu. Dôležitou z nich je nevykonanie zmeny žiadneho záznamu, ktorá je použitá pre rámce spracovávané inak ako v tabuľke tokov. Ďalšími sú nulovanie alebo nulovanie a export záznamu. Nulovanie znamená zápis počiatočných hodnôt do jednotlivých políčok záznamu. Export znamená rozšírenie UH o políčka záznamu (vytvorenie tokovej UH). Export a nulovanie záznamov o tokoch sa vykonáva pri skončení toku, ale často je potrebné realizovať ho aj v pravidelných intervaloch, aby softvérové aplikácie dostávali priebežné informácie o firmvérom monitorovaných tokoch. Realizácia riadenia pravidelného exportu používaných záznamov a správy voľných záznamov vo firmvéri je veľmi náročná, preto sa o obe bude starať kontrolný softvér SDM.

Exportér. Jednotka spolu páruje dvojice odpovedajúcich si vstupných UH záznamov a rámcov. Na základe im patriacej akcie potom rozhodne v akom formáte a ako cestou (DMA kanál) sú dáta poslané ďalej (do softvéru). Medzi podporované formáty pritom patrí: poslanie len UH (prípadne obohatenej o záznam toku), poslanie len celého paketu, zahodenie oboch.

Softvérový prístup. Predstavuje hlavný prístupový bod do SDM firmvéru zo softvéru. Obsahuje adresný dekodér, stavové a riadiace registre. Umožňuje priamy softvérový prístup k čítaniu položiek externej pamäte. Na základe softvérových príkazov je tiež schopná iniciovať pridávanie, odoberanie, upravovanie, exportovanie pravidiel a záznamov o tokoch.

Pamäťový arbiter. Zabezpečuje komunikáciu ostatných jednotiek s externou pamäťou. Medzi jeho základné zodpovednosti patrí: vhodné vzájomné prekladanie prístupov a správne smerovanie vyčítaných dát jednotkám. Dôležité je tiež zabezpečenie nezávislosti a atomicity operácií z vonkajšieho pohľadu. Teda hneď po sebe nasledujúce požiadavky na vykonanie viacerých operácií na rovnakej adrese vedú k jednoznačnému a očakávanému výsledku.

Bližší pohľad na fungovanie navrhovaného SDM firmvéru je možné získať aj z popisu základných operácií, ktoré bude realizovať. Jedná sa operácie: pridanie pravidla, odobranie pravidla, export a nulovanie záznamu o toku, klasifikácia rámca s úpravou záznamu o toku a klasifikácia rámca s exportom dát. Prvé tri z vymenovaných operácií iniciuje softvér cez jednotku softvérového prístupu, ďalšie dve iniciuje príchod rámca zo siete. Podrobnejšiemu popisu ich priebehu sa venuje nasledujúci text sekcie. Ilustrácie popisovaného priebehu jednotlivých operácií zakreslené priamo v schéme firmvéru z obrázku 5.1 je tiež možné nájsť v prílohe B.

Pridanie pravidla. Operácia začína naplnením konfiguračných registrov s jednotlivými zložkami pravidla a vyvolaním operácie pridania pravidla zo strany softvérového radiča SDM (krok 1). Jednotka softvérového prístupu na to zareaguje poslaním príkazu na pridanie pravidla klasifikátoru (krok 2). Klasifikátor vloží (ak je voľné miesto) implementačne závislým postupom nové pravidlo do tabuľky pravidiel (krok 3). V prípade, že akcia vkladaného pravidlo pracuje s istým záznamom o toku v tabuľke 2 je zaslaná požiadavka na jeho vynulovanie správcovi záznamov (krok 4). Ten zapíše do daného záznamu počiatočnú hodnotu (krok 5). Popísaný postup je ilustrovaný na obrázku B.1.

Odobranie pravidla. Operácia začína naplnením konfiguračných registrov s jednotlivými zložkami identifikátoru pravidla (akciu netreba vypĺňať) a vyvolaním operácie zrušenia pravidla zo strany softvérového radiča SDM (krok 1). Jednotka softvérového prístupu na to zareaguje poslaním príkazu na odstránenie pravidla klasifikátoru (krok 2). Klasifikátor zruší platnosť pravidla (ak také existuje) v tabuľke pravidiel (krok 3). V prípade, že prečítaná akcia rušeného pravidla (krok 4) pracuje s istým záznamom o toku v tabuľke 2 je zaslaná požiadavka na jeho export správcovi záznamov (krok 5). Požiadavka nesie UH hlavičku s vyplneným identifikátorom toku. Správca záznamov načíta polia daného záznamu z tabuľky 2 (kroky 6 a 7) a rozšíri nimi UH hlavičku (vznikne tokovú UH). Tá ďalej pokračuje do exportéru (krok 7), ktorý ju priamo pošle dátovou cestou do softvéru (krok 8). Z FIFO pamäte pritom nie je načítaný žiaden rámec. Export tokovej UH je možné podmieniť jej porovnaním s počiatočnou hodnotou záznamu o toku, kedy nezmenenú tokovú UH (neprišiel pre ňu žiaden rámec) nie je potrebné exportovať. Popísaný postup je ilustrovaný na obrázku B.2.

Export a nulovanie záznamu o toku. Operácia začína naplnením konfiguračných registrov s jednotlivými zložkami identifikátoru pravidla (akciu netreba vypĺňať) a vyvolaním operácie exportu pravidla zo strany softvérového radiča SDM (krok 1), ktorý tak robí v pravidelných intervaloch pre jednotlivé pravidlá. Jednotka softvérového prístupu na to zareaguje poslaním príkazu na export pravidla klasifikátoru (krok 2). Klasifikátor prečíta akciu pravidla (ak také existuje) v tabuľke 1 (kroky 3 a 4). Pokračuje zaslaním požiadavky na export z akcie zisteného záznamu správcovi záznamov (krok 5). Požiadavka nesie UH hlavičku s vyplneným identifikátorom toku. Správca záznamov vyčíta a následne vynuluje

polia daného záznamu z tabuľky 2 (kroky 6 a 7). Prečítanými hodnotami rozšíri UH hlavičku (vznikne tokovú UH). Tá ďalej pokračuje do exportéru (krok 7), ktorý ju priamo pošle dátovou cestou do softvéru (krok 8). Z FIFO pamäte pritom nie je vyčítaný žiaden rámec. Podobne ako v predošlej operácii je export tokovej UH možné podmieniť jej porovnaním s počiatočnou hodnotou záznamu o toku. Popísaný postup je ilustrovaný na obrázku B.3.

Klasifikácia rámca s úpravou záznamu o toku. Operáciu vyvolá prijatie rámca zo sieťovej linky a jeho doručenie na vstup SDM firmvéru (krok 1). Dáta vstupného rámca sú na dobu spracovania odložené do FIFO pamäte (krok 2). Paralelne s odkladaním začína analýza hlavičiek a extrakcia zaujímavých informácií z nich. Extrahované položky sú vo forme UH predané klasifikátoru (krok 3). Ten sa pokúsi klasifikovať prijatú UH podľa aktuálne nakonfigurovaných pravidiel. Musí kvôli tomu pristúpiť do externej pamäte (kroky 4 a 5). Výsledkom klasifikácie je akcia definujúca ďalšie spracovanie. Spolu s UH pokračujú do správcu záznamov (krok 6). Keďže v tomto prípade bola nájdenou akciou úprava istého záznamu o toku, správca záznamov túto položku načíta (kroky 7 a 8), upraví a zapíše späť (krok 9). Navyše ešte zašle exportéru požiadavku na odstránenie rámca z FIFO pamäte a jeho zahodenie (krok 10). Z FIFO pamäte sú teda postupne načítané slová prvého rámca (krok 11) a sú zahadzované. Popísaný postup je ilustrovaný na obrázku B.4.

Klasifikácia rámca s exportom dát. Operácia sa až po príchod do správcu záznamov (krok 6) chová presne rovnako ako predošlá popísaná. Zmena nastáva, pretože v tomto prípade nebola nájdenou akciou úprava istého záznamu o toku, ale istý druh exportu dát do softvéru. Správca záznamov, tak nemusí pristupovať do tabuľky záznamov. Musí však zabezpečiť synchronizáciu s rozpracovanými operáciami (krok 7), aby nedošlo k ich vzájomnému predchádzaniu. Inak UH hlavičku aj s akciou zasiela bez zmeny exportéru (krok 8). Ten postupne vyberá z FIFO pamäte slová prvého rámca (krok 9) a ďalej s nimi pracuje podľa prijatej akcie. V prípade, že je vybraný export celého rámca sú jeho slová exportované do softvéru a UH je zahodená (krok 10). V prípade, že je zvolený export UH hlavičky sú slová rámca zahadzované a exportovaná do softvéru je UH (krok 10). Nakoniec ak je zvolené úplné zahodenie nie je nič exportované a UH aj rámec sú zahodené (nevykoná sa krok 10). Popísaný postup je ilustrovaný na obrázku B.5.

5.2 Softvérová časť

Jadrom softvérovej časti SDM architektúry je SDM radič realizujúci riadenie predspracovania dátového toku z vysokorýchlostnej siete firmvérom na základe požiadaviek od užívateľských aplikácií. Radič komunikuje s firmvérom len pomocou riadiacej cesty, ale nijako nespravuje ani priamo nezasahuje do dátovej cesty vedúcej z neho. Očakáva, že funkcionalitu spojenú s prenosom predspracovaných sieťových dát medzi hardvérom a softvérom realizujú ovládače platformy, nad ktorou funguje (napr. SZE pre NetCOPE), alebo iný na to špeciálne určený softvér. SDM radič teda predstavuje hlavný riadiaci prvok celého SDM systému. Zároveň poskytuje užívateľským aplikáciám formou istého jednoduchého rozhrania možnosť podieľať sa na ovládaní predspracovania sieťových dát podľa ich vlastných potrieb. Užívateľskou aplikáciou je pritom chápaný ľubovoľný program schopný komunikácie s radičom. Funkcionalitu požadovanú od SDM radiča je tak možné rozdeliť do dvoch základných skupín: operácie súvisiace priamo s ovládaním SDM firmvéru a operácie zabezpečujúce softvérový prístup aplikácií k riadeniu.

SDM radič pri realizácii všetkých operácií s firmvérom SDM komunikuje riadiacou cestou priamo s jednotkou softvérového prístupu. Pre potreby správneho ovládania funkcionality SDM firmvéru musí radič vykonávať hlavne nasledujúce operácie:

Správa pravidiel. Radič sa stará o zmenu pravidiel definujúcich spôsob predspracovania jednotlivých tokov vo firmvéri na základe požiadaviek od aplikácií. Musí byť preto schopný iniciovať a prípadne aj do istej miery kontrolovať proces samotného pridávania, odoberania a zmeny konkrétnych pravidiel vo firmvéri SDM.

Pravidelný export záznamov o tokoch. Jednotka správy tokov v SDM firmvéri priamo podporuje operácie na export a správu záznamov o tokoch. Avšak nie je samočinne schopná pravidelne ich exportovať do softvéru. Export musí pre jednotlivé záznamy iniciovať v pravidelných intervaloch samotný SDM radič.

Sledovanie obsadenosti položiek tabuľky so záznamami o tokoch. Jednotka správy tokov v SDM firmvéri pracuje so záznamami o tokoch na základe akcie prijatej od klasifikátoru. Akcia je získaná ako výsledok klasifikácie rámcov podľa radičom nastavených pravidiel a mimo iného určuje aj adresu konkrétneho záznamu o toku, na ktorý sa vzťahuje. SDM radič preto pri pridávaní nových pravidiel potrebuje poznať, ktoré záznamy sú aktuálne voľné a môžu byť použité. Radič si teda musí uchovávať informácie o obsadených a voľných miestach v tabuľke záznamov o tokoch.

SDM radič reaguje na požiadavky od viacerých užívateľských aplikácií (procesov) ohľadom použitého spôsobu predspracovania a vhodne ich agreguje a aplikuje do firmvéru. Agregácia vychádza z toho, že spôsoby predspracovania je možné rozdeliť do rôznych stupňov podľa miery zachovávanej informácie o rámcoch. Po rade od najnižšieho stupňa (najslabšieho) predspracovania ide o:

Žiadne: rámce sú posielané celé bez zmeny

Čiastočné: extrahované a posielané sú len zaujímavé informácie z rámcov vo forme UH

Úplné: informácie o rámcoch sú firmvérom agregované do záznamu o toku

Zahadzovanie: úplne všetky informácie o rámcoch sú ignorované

Vyšší stupeň predspracovania znamená posielanie menej presnej informácie do softvéru. Agregácia spočíva teda vo výbere najnižšieho stupňa predspracovania daného toku (zachovanie najviac informácií) zo všetkých platných kontextov.

Po predstavení kontextov a spôsobu ich agregácie ja možné bližšie spresniť funkcionalitu SDM radiča potrebnú na umožnenie prístupu softvérových aplikácií k riadeniu. Jedná sa najmä o nasledujúce operácie:

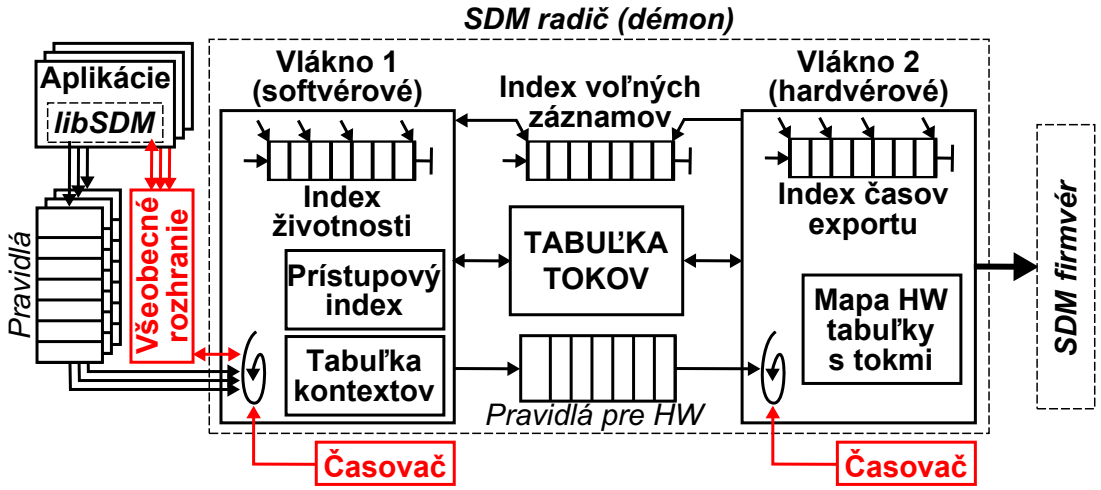
Správa kontextov. Aplikácie musia mať možnosť tvoriť, rušiť a upravovať kontexty predspracovania dát. Jednotlivé kontexty musia byť navzájom jednoznačne rozlíšiteľné (napr. číslom), aby bolo možné ďalej s nimi pracovať (upravovať pravidlá v nich). Každému z nich môže byť zároveň priradená množina nemenných pravidiel o skupinách tokov definujúcich základné správanie sa k novým tokom.

Správa zdrojov pravidiel. Jednotlivé aplikácie vytvárajúce pravidlá pre SDM ich potrebujú nejakým spôsobom odovzdať SDM radiču. Radič preto musí poskytovať operácie na zaregistrovanie a odregistrovanie zdrojov pravidiel. Zároveň musí vedieť prijímať pravidlá zo všetkých aktuálne registrovaných zdrojov.

Správa pravidiel v kontextoch. Aplikácie registrované ako zdroje dát majú možnosť pridávať/rušiť pravidlá na zvýšenie stupňa predspracovania jednotlivých tokov v SDM

firmvéri v existujúcich kontextoch. Radič musí vedieť tieto pravidlá prijímať, udržovať, agregovať a následne aplikovať do firmvéru. Pridávanie a rušenie upresňujúcich pravidiel v kontextoch predstavuje najfrekventovanejšie využívanú operáciu SDM radiča.

Časová expirácia pravidiel. Pridávané pravidlá v jednotlivých kontextoch nemôžu mať nekonečnú platnosť. Platnosť pravidiel síce môže skončiť implicitnou požiadavkou od samotnej aplikácie na ich zrušenie, ale spoliehať sa na takéto zrušenie všetkých zavedených pravidiel nie je vždy možné alebo praktické. Vhodné je preto zaviesť maximálnu časovú platnosť pridávaných pravidiel a najneskôr po vypršaní tejto platnosti ich automaticky rušiť.



Obr. 5.2: Celková schéma softvérového radiča SDM

Schéma celkového návrhu SDM radiča je zobrazená na obrázku 5.2. Radič je realizovaný formou démonizovaného procesu bežiaceho na pozadí. Aplikácie sú schopné komunikovať s démonom pomocou rôznych operácií implementovaných formou knižnice (libSDM). Základ komunikácie je postavený na využití vhodného spôsobu medziprocesnej komunikácie z možností poskytovaných operačným systémom. Na zobrazenej schéme je vidno rozdelenie funkcionality SDM radiča na dve vlákna. Prvé z nich realizuje len operácie spojené s príjmom, spracovaním a agregáciou pravidiel od softvérových aplikácií. Cyklicky prechádza jednotlivé zdroje pravidiel (čierne) a ďalších udalostí a získava tak príkazy, ktoré realizuje. Druhé vlákno sa stará výlučne o operácie spojené s konfiguráciou s firmvérom SDM. Na základe inštrukcií od prvého vlákna pridáva alebo ruší pravidlá z firmvéru. Zároveň na základe vypršania časovača iniciuje pravidelný export záznamov o tokoch. Funkcionalita oboch vlákien môže byť samozrejme realizovaná aj v rámci jediného vlákna, ale uvedeným rozdelením je možné dosiahnuť vyššiu efektivitu riešenia. Prvé vlákno totiž môže spracúvať nové pravidlá od aplikácií a chystať druhému vláknu záznamy pre firmvér, zatiaľ čo druhé vlákno realizuje potrebné operácie s firmvérom. Tým je možné dosiahnuť plne saturované využitie riadiacej cesty do firmvéru.

Základnou dátovou štruktúrou na uchovanie informácií (pravidiel) od aplikácií je tabuľka tokov (v strede) zdieľaná medzi vláknami radiča. V tabuľke sú uložené záznamy pre všetky toky, o ktorých má radič v danom čase nejakú doplňujúcu informáciu k ich spracovaniu. Teda ide o všetky toky, ktorých pravidlá sú aktuálne vo firmvéri a aj o toky, ktoré len čakajú

na možné doplnenie informácie a následné vloženie do firmvéru. Tabuľka tokov je realizovaná jednoduchým poľom záznamov o tokoch. Každý záznam je tak unikátne identifikovateľný svojim číselným indexom (pozíciou) v tomto poli. Na efektívnejšiu prácu a rýchlejší prístup k záznamom je nad tabuľkou tokov potrebné vytvoriť niekoľko ďalších pomocných indexov. Jedná sa o nasledujúce indexy:

Prístupový index: slúži na urýchlenie vyhľadávania záznamov podľa identifikátora toku. Index pokrýva všetky aktuálne obsadené položky tabuľky. Realizovaný môže byť napríklad implicitne pomocou haš funkcie alebo usporiadaním záznamov do istého druhu vyvažovaného stromu (AVL, RB ...).

Index voľných záznamov: uchováva množinu nezabraných položiek v tabuľke tokov. V prípade použitia implicitného prístupového indexu (napr. haš funkcia) je tiež implicitne daný. V ostatných prípadoch je realizovateľný jednoduchým zoznamom.

Index životnosti: umožňuje jednoduchú identifikáciu záznamov, ktorým vypršala doba platnosti. Obsahuje všetky aktuálne platné záznamy z tabuľky zoradené vzostupne podľa času skončenia ich platnosti. Realizovateľný je dvojsmerne viazaným zoznamom dodatočne rozdeleným na časové bloky položiek, aby sa znížila potrebná presnosť expiračného časovača (expiruje sa po celých blokoch, nie po jednotlivých záznamoch). Každý novovytvorený záznam v tabuľke je pridaný na koniec zoznamu (do posledného bloku) a v prípade vypršania časovača je zoznam posunutý o blok vpred. Teda záznamy prvého bloku sú odstránené z tabuľky tokov a nové záznamy sú následne vkladané do nového bloku na konci zoznamu.

Index časov exportu: pokrýva záznamy nahrané vo firmvéri SDM, ktoré pracujú s položkou tabuľky záznamov o tokoch (tabuľka2). Realizácia sa podobá vyššie popísanému indexu životnosti, teda ide o vzostupne zoradenú postupnosť záznamov rozdelenú na bloky. Zoradené sú podľa času pridania alebo času posledného exportu. Významným rozdielom je, že index exportov predstavuje cyklický zoznam. Teda v prípade posunu o blok pri vypršaní časovača sú zaslané požiadavky na export všetkých tokov z prvého bloku a tento blok je pripojený na koniec zoznamu bez zrušenia jeho tokov. Docielené je tak pravidelné opakovanie sa požiadaviek na export pre jednotlivé záznamy.

Okrem tabuľky tokov sú ukladané ešte ďalšie dôležité informácie. Prvé vlákno (softvérové) spracúva zoznam zdrojov pravidiel, ktoré sú pravidelne testované na novoprichádzajúce záznamy a tabuľku aktuálne definovaných kontextov. Pridávanie a odoberanie zdrojov pravidiel aj kontextov je možné cez všeobecné rozhranie SDM radiča, ktoré je mapované v zdieľanej pamäti s dohodnutým kľúčom. Uvedené rozhranie má formu požiadavka-odpoveď, kedy aplikácie zapíšu svoju požiadavky a čakajú na jej vybavenie a zapísanie odpovede zo strany SDM radiča. Prístup k popísanému rozhraniu zo strany aplikácií musí byť výlučný, aby sa ich požiadavky navzájom neprepisovali. Druhé vlákno (hardvérové) pracuje s mapou zaplnenia firmvérovej tabuľky2 (záznamy o tokoch). Potrebuje ukladať zoznam prázdnych položiek, kvôli prideľovaniu novým záznamom a prideleniu zaplnených položiek jednotlivým pravidlám, kvôli správnemu uvoľňovaniu pri rušení pravidiel.

Bližší náhľad na fungovanie navrhnutého SDM radiča je možné získať aj z popisu základných operácií, ktoré bude realizovať. Ide o operácie: pridanie/zrušenie kontextu, pridanie/zrušenie zdroja informácií, pridanie/úprava/zrušenie pravidla, vypršanie platnosti záznamov a vyžiadanie exportu záznamov z firmvéru. Prvé tri z vymenovaných operácií iniciujú softvérové aplikácie cez všeobecné rozhranie SDM radiča alebo cez jedno jeho vstupné

rozhranie. Ďalšie dve iniciuje vypršanie časovača, vlákna môžu využívať jeden spoločný časovač alebo každé z nich jeden svoj vlastný. Podrobnejšiemu popisu priebehu uvedených operácií sa venuje nasledujúci text sekcie. Ilustrácie jednotlivých popisovaných operácií zakreslené priamo v schéme SDM radiča z obrázku 5.2 je tiež možné nájsť v prílohe C.

Pridanie/zrušenie kontextu. Operácia začína vytvorením požiadavku na zmenu kontextov (pridanie/zrušenie) od aplikácie pomocou volania funkcie SDM knižnice. Požiadavka sa zapíše do všeobecného rozhrania SDM radiča (krok 1). Softvérové vlákno následne objaví a načíta nevybavenú požiadavku vo všeobecnom rozhraní (krok 2). Požadovaná zmena v kontextoch je premietnutá do tabuľky kontextov (ak je možná) a informácia o úspešnosti operácie je vložená ako odpoveď do všeobecného rozhrania a aplikácii je oznámená dokončenie operácie (krok 3). V prípade, že ide o tvorbu nového kontextu je odpoveď doplnená aj o číslo vytvoreného kontextu. Po skončení operácie je aplikácii vrátená jej úspešnosť a výsledok (krok 4). Úspešná zmena v kontextoch (krok 3) vyvolá paralelne s odpoveďou aplikácii aj odoslanie správy pre hardvérové vlákno o zmene v kontextoch (krok 5). Po prijatí správy o zmene kontextov (krok 6) sa hardvérové vlákno postará o jej premietnutie do firmvéru SMD (krok 7). Popísaný postup je ilustrovaný na obrázku C.1. Pri zmene v kontextoch môže byť ešte vhodné zabezpečiť zrušenie všetkých aktuálne definovaných pravidiel.

Pridanie zdroja informácií. Operácia začína vytvorením požiadavky na pridanie zdroja informácií od aplikácie pomocou volania zodpovedajúcej funkcie SDM knižnice. Požiadavka sa zapíše do všeobecného rozhrania SDM radiča (krok 1). Softvérové vlákno následne objaví a načíta nevybavenú požiadavku vo všeobecnom rozhraní (krok 2). Pre nový zdroj informácií je vytvorený nový prostriedok medziprocesovej komunikácie (krok 3) a jeho identifikácia je zapísaná ako odpoveď do obecného rozhrania radiča a aplikácii je oznámený koniec operácie (krok 4). Po skončení operácie je aplikácii vrátená zapisovacia strana vytvoreného komunikačného prostriedku (krok 5). Popísaný postup je ilustrovaný na obrázku C.2.

Zrušenie zdroja informácií. Operácia začína vytvorením požiadavku na zastavenie komunikácie so SDM radičom od aplikácie pomocou volania odpovedajúcej funkcie SDM knižnice. Požiadavka sa zapíše do obecného rozhrania SDM radiča (krok 1). Softvérové vlákno následne objaví a načíta nevybavenú požiadavku v obecnom rozhraní (krok 2). Prostriedok medizprocesovej komunikácie patriaci rušenému zdroju je uvoľnený (krok 3) a aplikácii je oznámený koniec operácie (krok 4). Po skončení operácie (krok 5) pokračuje aplikácia ďalej vo svojej práci. Popísaný postup je ilustrovaný na obrázku C.3.

Pridanie pravidla. Operácia je iniciovaná aplikáciou cez volanie funkcie SDM knižnice na pridanie pravidla. Pridávané pravidlo je odoslané komunikačným prostriedkom radiču SDM (krok 1). Softvérové vlákno ho po čase prečíta (krok 2) a zaháji jeho pridávanie. Začína sa prístupom do tabuľky kontextov (krok 3), kde sa vytvorí základ záznamu do tabuľky tokov podľa kontextov zodpovedajúcich toku pravidla. Následne je pomocou prístupového indexu vyhľadané, či pre tok pridávaného pravidla už neexistuje záznam (krok 4). V tomto prípade nebude záznam nájdený, preto sa alokuje nový pomocou indexu voľných záznamov (krok 5). Novo vytvorený záznam je potom postupne: vložený na koniec indexu životnosti (krok 6), správne vyplnený v tabuľke tokov (krok 7) a zapojený do prístupového indexu (krok 8). V prípade, že vkladané pravidlo môže byť priamo vložené do firmvéru, odošle softvérové vlákno požiadavku na jeho vloženie hardvérovému (krok 9). Reakcia na prijatie požiadavky vloženia pravidla hardvérovým vláknom (krok 10) je zabezpečenie operácií súvisiacich s jeho vložením do firmvéru (krok 11). Navyše, ak ide o pravidlo s akciou pracujúcou so záznamom o toku vo firmvéri, je potrebné alokovať mu jeden takýto záznam pomocou

mapy HW tabuľky s tokmi (krok 13) a zabezpečiť jeho pravidelný export vložením na koniec indexu exportov (krok 12). Popísaný postup je ilustrovaný na obrázku C.4.

Úprava pravidla. Operácia začína úplne rovnako ako vyššie popísané pridávanie pravidla. Rozdiel nastáva až pri vyhľadaní existencie záznamu pre tok pravidla pomocou prístupového indexu (krok 4). V tomto prípade bude záznam nájdený, preto nie je potrebné alokovať nový. Stačí agregovať novo pridávané pravidlo s existujúcim záznamom v tabuľke tokov (krok 5). V prípade, že agregáciou je dosiahnuté pravidlo, ktoré môže byť priamo vložené do firmvéru, odošle softvérové vlákno požiadavku na jeho vloženie hardvérovému (krok 6). Hardvérové vlákno spracúva pridanie pravidla podobne ako je popísané v predošlom odseku. Rozdiel je len v tom, že pridávané pravidlo už môže byť vo firmvéri uložené s inou akciou, ktorú treba korektne prepísať. Popísaný postup je ilustrovaný na obrázku C.5.

Zrušenia pravidla. Operácia je iniciovaná aplikáciou cez volanie funkcie SDM knižnice na zrušenie pravidla pre daný tok zo SDM. Rušený tok je odoslaný komunikačným prostriedkom radiču SDM (krok 1). Softvérové vlákno ho po čase prečíta (krok 2) a zaháji rušenie. Na začiatku je pomocou prístupového indexu (krok 3) nájdený zodpovedajúci záznam v tabuľke tokov a zároveň je zrušený z prístupového indexu. Záznam je potom postupne: označený za neplatný (krok 4), vložený do indexu voľných záznamov (krok 5) a zrušený z indexu životnosti. V prípade, že vkladané pravidlo je vložené vo firmvéri, odošle softvérové vlákno požiadavku na jeho odstránenie aj hardvérovému (krok 7). Reakciou na prijatie požiadavky zrušenia pravidla hardvérovým vláknom (krok 8) je hlavne vykonanie operácií súvisiacich s jeho odstránením z firmvéru (krok 9). Navyše, ak ide o pravidlo s akciou pracujúcou so záznamom o toku vo firmvéri je potrebné uvoľniť jemu patriaci záznam v mape HW tabuľky s tokmi (krok 11) a zrušiť jeho pravidelný export odstránením z indexu exportov (krok 10). Popísaný postup je ilustrovaný na obrázku C.6.

Vypršanie časovača platnosti. Operáciu začína vypršanie časovača spojeného s určovaním konca životnosti záznamov tabuľky tokov (krok 1). Vypršanie časovača je následne zachytené softvérovým vláknom (krok 2) a začína samotný proces expirovania tokov. Realizované je postupne: zrušením platnosti záznamu v tabuľke tokov (krok 3), vložením rušenej položky na začiatok indexu voľných záznamov (krok 4) a odstránením z prístupového indexu (krok 5). Expirované a rušené sú len záznamy tokov odkazované prvým blokom indexu životnosti, ktorý je po dokončení zrušený (krok 6). V prípade rušených záznamov, ktoré sú vložené vo firmvéri, odošle softvérové vlákno navyše požiadavky na ich zrušenie hardvérovému (krok 7). Hardvérové vlákno potom spracúva rušenie pravidla rovnako ako je popísané v predošlom odseku. Popísaný postup je ilustrovaný na obrázku C.7.

Vypršanie časovača exportu. Ako sám názov napovedá, operácia začína vypršaním časovača spojeného s exportom tokov z firmvéru SDM (krok 1). Vypršanie časovača je následne zachytené hardvérovým vláknom (krok 2) a začína samotný proces vyvolania exportu tokov. Realizované je zasielaním požiadaviek na export záznamov jednotlivých tokov do firmvéru SDM (krok 3). Exportované sú len toky odkazované prvým blokom indexu exportov, ktorý je po dokončení presunutý na koniec indexu (krok 4), čím je zabezpečené opakované a pravidelné volanie exportu jednotlivých tokov z firmvéru. Popísaný postup je ilustrovaný na obrázku C.8.

Kapitola 6

Implementácia

Kapitola popisuje základné vlastnosti konkrétnych vytvorených implementácií súčastí SDM systému vychádzajúce z návrhu v predošlej kapitole. Prvé štyri sekcie sa venujú implementácii softvérových častí SDM potrebných na vytvorenie simulačného modelu, ktorý je jedným z požadovaných výstupov tejto práce. Implementácie sú vytvárané v prostredí operačného systému Linux. Prvá sekcia sa venuje testovaniu výkonnostných parametrov rôznych spôsobov medziprocesovej a medzivláknovej komunikácie dostupných v operačnom systéme Linux s cieľom nájdenia najlepšieho z nich pre potreby softvérovej časti SDM. Nasledujúca sekcia popisuje vytvorenú knižnicu libSDM implementujúcu detaily komunikácie medzi užívateľskými aplikáciami a SDM radičom. Nasleduje bližší popis implementácie samotného SDM radiča a jeho nastavovania pri spustení. Poslednou zo spomínaných štyroch sekcií je podrobný popis realizácie simulačného modelu systému SDM.

Posledné dve sekcie kapitoly sa venujú nad rámec zadania rozriešeným problémom konkrétnej realizácie dvoch dôležitých jednotiek SDM firmvéru. Predposledná sekcia kapitoly popisuje dokončenú finálnu implementáciu modulárneho analyzátora hlavičiek rámcov. Posledná sekcia popisuje vytvorený prototyp hlavnej časti jednotky klasifikátora rámcov na základe pravidiel o tokov. Prototyp realizuje klasifikáciu využitím haš tabuliek a princípu kukučieho hašovania.

6.1 Medziprocesová a medzivláknová komunikácia

Jednou z najdôležitejších úloh pri realizácii softvérovej časti SDM systému je výber vhodného spôsobu realizácie komunikácie medzi užívateľskými aplikáciami (procesmi) a SDM radičom, a tiež medzi obomi vláknami samotného SDM radiča. Konkrétne ide o výber vhodného spôsobu medziprocesovej resp. medzivláknovej komunikácie z možností ponúkaných operačným systémom. V Unixe existuje niekoľko rôznych spôsobov realizácie medziprocesovej komunikácie [27]. Pre efektivitu SDM radiča je potrebné vybrať ten z nich, ktorý ponúka čo najvyššiu rýchlosť resp. čo najmenšiu réžiu. Na vyhodnotenie rýchlostí dosahovaných jednotlivými spôsobmi, a teda aj vhodnosti ich použitia, bol implementovaný jednoduchý testovací program.

Hlavné telo testovacieho programu je spoločné pre všetky posudzované spôsoby komunikácie. Jeho fungovanie je možné zapísať pseudokódom uvedeným v algoritme 1. Kde W označuje pri preklade nastaviteľný počet producentov (pôvodcov správ) a M nastaviteľný počet správ, ktoré každý producent vytvorí a odošle. Okrem toho je možné nastaviť dĺžku prenášaných správ (D) a či konzumenta (príjemcu správ) a producentov spúšťať ako

nové vlákna alebo ako nové procesy (funkcia start). Samotný priebeh testu začína (funkcia main) vytvorením požadovaného počtu producentov, reprezentujúcich užívateľské aplikácie a jedného konzumenta, predstavujúceho SDM radič. Beh každého producenta (funkcia producent) začína inicializáciou komunikácie s konzumentom, pokračuje vytvorením (funkcia produkuj) a odoslaním (funkcia pošli) M správ konzumentovi. Vytvorenie správy je realizované vygenerovaním D bajtov pseudonáhodným generátorom. Po odoslaní požadovaného počtu správ je komunikácia korektne ukončená a producent končí. Beh konzumenta (funkcia konzument) začína inicializáciou komunikácií s jednotlivými producentmi, pokračuje príjmom (funkcia prijmi) a spracovaním správy (funkcia konzumuj). Spracovanie správy je realizované vygenerovaním D bajtov rovnakým pseudonáhodným spôsobom ako producenti a ich porovnaním s prijímanými správami na zhodu. Po prijatí všetkých správ sú komunikácie korektne ukončené a konzument končí. Funkcie na inicializáciu a ukončovanie komunikácií aj funkcie na odosielanie a príjem správ sú implementované špecificky podľa potrieb jednotlivých spôsobov medziprocesovej komunikácie. Pri preklade testovacieho programu je potom možné voliť, ktorý konkrétny spôsob implementácie má byť použitý.

Algoritmus 1: Základný test komunikácie

```
1  main:
2      for 1 to W do
3          spusti(producent)
4      spusti(konzument)

5  producent:
6      inicializuj komunikáciu s konzumentom
7      for 1 to M do
8          o = produkuj()
9          pošli(o)
10     ukonči komunikáciu

11 konzument:
12     inicializuj komunikácie s producentmi
13     for 1 to M × W do
14         o = prijmi()
15         konzumuj(o)
16     ukonči komunikácie
```

Z rôznych spôsobov realizácie medziprocesovej komunikácie dostupných v Unixe s využitím jazyka C bolo vybraných niekoľko najvhodnejších pre potreby SDM softvéru, ktoré boli podrobené popísanému testu výkonnosti. Konkrétne ide o nasledujúce spôsoby [27]:

Pomenované rúry: predstavujú najzákladnejší a najľahší spôsob medziprocesovej komunikácie v Unixe. Okrem pomenovaných existujú aj nepomenované rúry, s ktorými sa je možné najčastejšie stretnúť pri použití operátora `|` v príkazovom riadku. Problémom nepomenovaných rúr je však obmedzenie ich použitia len na komunikáciu navzájom súvisiacich procesov (rodič a potomkovia). Rúry všeobecne fungujú ako FIFO zoznamy, do ktorých je možné z jednej strany zapisovať dáta a na druhej ich v rovnakom poradí načítať. Zápis dát je realizovaný pomocou štandardnej funkcie `write()`, ktorá má zaručenú atomicitu do istej veľkosti zapisovaných dát [28]. Komunikáciu

viacerých producentov s jedným konzumentom pomocou pomenovaných rúr je tak možné realizovať vlastnou rúrou pre každého producenta aj jedinou spoločnou rúrou pre všetkých.

Fronty správ: pracujú podobne ako pomenované rúry, ale navyše majú niektoré dodatočné vlastnosti. Dáta sú pri práci s nimi združované do formy správ istého typu a dĺžky. Pri zápise sú do fronty vkladané jednotlivé správy za sebou a pri štandardnom čítaní sú z nej v rovnakom poradí vyberané. Navyše je ale možné realizovať načítanie konkrétnych typov správ predčasne. Zápisy aj načítania správ z fronty sú atomické, preto je testovanú úlohu možné riešiť s použitím vlastnej fronty správ pre každého producenta aj jedinej spoločnej pre všetkých producentov.

Sokety: umožňujú vytvárať obojsmerné spojenia komunikujúcich dvojíc. Je možné využiť ich na komunikáciu v rôznych doménach. Známe a najčastejšie je ich použitie v doméne umožňujúcej sieťovú komunikáciu. V Unixe však existuje aj doména umožňujúca využiť sokety na obojsmernú komunikáciu dvojíc procesov. Keďže sokety podporujú len spojenia typu jeden s jedným, je možné ich využiť len vo variante s jedným soketom pre každého producenta.

Zdieľané pamäťové segmenty: sú segmenty pamäte zdieľané medzi procesmi. Zdieľaný segment je po vytvorení možné jednoducho zapojiť do logického adresného priestoru niekoľkých procesov zároveň. Zmeny vykonané v tomto segmente sú potom viditeľné pre všetky k nemu pripojené procesy. Pre potreby komunikácie v SDM je nad zdieľanou pamäťou potrebné implementovať istú formu vlastného FIFO zoznamu správ (dát). Najjednoduchší spôsob implementácia FIFO zoznamu pracuje so statickým poľom položiek a ukazovateľmi na začiatok a koniec aktuálne zaplnenej časti v ňom. Táto realizácia umožňuje, pri použití nad zdieľanou pamäťou, jednosmernú komunikáciu dvojice procesov. Použiteľná je teda vo variante s jedným zoznamom pre každého producenta. Pri realizácii zdieľaného FIFO zoznamu je potrebné vhodne ošetriť súbežný prístup k ukazovateľom začiatku a konca. Je možné pritom použiť kľúčové slovo `volatile` jazyka C, čo by malo byť dostatočné na platformách s procesorom rodiny x86 alebo x64, ale všeobecne je to nedostatočné riešenie [29]. Všeobecnejšie riešenie predstavuje využitie vstavaných funkcií atomického prístupu k pamäti ponúkaných prekladačom `gcc` [30].

Spôsob komunikácie	**Počet producentov**				
	1	**2**	**3**	**4**	**6**
Pomenované rúry	7,258 s	14,345 s	21,643 s	29,173 s	44,993 s
Pomenovaná rúra (spoločná)	7,394 s	15,761 s	27,155 s	48,738 s	132,663 s
Fronty správ	10,123 s	34,543 s	72,807 s	94,918 s	137,199 s
Fronta správ (spoločná)	9,574 s	40,715 s	100,231 s	165,598 s	369,871 s
Sokety	10,259 s	21,807 s	31,416 s	43,614 s	67,799 s
Zdieľané FIFO (volatile)	2,861 s	4,112 s	5,971 s	7,738 s	11,768 s
Zdieľané FIFO (gcc atomic)	3,268 s	4,763 s	6,648 s	8,399 s	12,814 s

Tabuľka 6.1: Namerané doby behu pri rastúcej záťaži a použití procesov

Samotné testovanie prebiehalo na počítači s operačným systémom Linux, konkrétne verzia Scientific Linux 6.3 s jadrom verzie 2.6.32. Počítač bol vybavený štvorjadrovým

Spôsob komunikácie	Počet producentov				
	1	**2**	**3**	**4**	**6**
Pomenované rúry	7,790 s	16,744 s	24,096 s	31,906 s	49,263 s
Pomenovaná rúra (spoločná)	7,576 s	16,936 s	29,811 s	46,712 s	138,242 s
Fronty správ	10,183 s	33,727 s	74,946 s	95,644 s	139,477 s
Fronta správ (spoločná)	9,816 s	39,814 s	90,759 s	167,455 s	368,191 s
Sokety	10,659 s	23,783 s	34,255 s	46,350 s	72,084 s
Zdieľané FIFO (volatile)	2,865 s	4,149 s	6,562 s	8,069 s	11,665 s
Zdieľané FIFO (gcc atomic)	3,285 s	4,715 s	7,219 s	8,932 s	13,238 s

Tabuľka 6.2: Namerané doby behu pri rastúcej záťaži a použití vlákien

Spôsob komunikácie	Počet producentov				
	1	**2**	**3**	**4**	**6**
Pomenované rúry	7,044 s	7,190 s	7,223 s	7,316 s	7,474 s
Pomenovaná rúra (spoločná)	7,182 s	7,800 s	9,827 s	11,894 s	15,756 s
Fronty správ	10,123 s	17,217 s	23,634 s	23,688 s	22,815 s
Fronta správ (spoločná)	9,442 s	19,110 s	34,011 s	41,889 s	63,059 s
Sokety	10,044 s	10,003 s	10,693 s	10,870 s	11,428 s
Zdieľané FIFO (volatile)	2,860 s	2,057 s	1,991 s	1,958 s	1,988 s
Zdieľané FIFO (gcc atomic)	3,268 s	2,364 s	2,209 s	2,113 s	2,149 s

Tabuľka 6.3: Namerané doby behu pri konštantnej záťaži a použití procesov

procesorom Intel Xeon E5420 s frekvenciou 2,5 GHz. Na meranie doby trvania testov bol využitý program `time`, ktorým bol meraný reálny čas behu programu. Pri testovaní bola použitá dĺžka správy $D = 48$ bajtov a rôzne počty producentov ($W \in \{1, 2, 3, 4, 6\}$). Počet produkovaných a následne zasielaných správ bol $p = 10\,000\,000$. Výsledky testov pre jednotlivé počty producentov pri rastúcej celkovej záťaži ($M = p$) a použití procesov sú uvedené v tabuľke 6.1. Výsledky rovnakých testov s využitím vlákien namiesto procesov sú zachytené v tabuľke 6.2. Nakoniec boli ešte realizované testy pre jednotlivé počty producentov pri konštantnej celkovej záťaži ($M = p/W$) a použití procesov s výsledkami zachytenými v tabuľke 6.3.

Z nameraných výsledkov uvedených vo vyššie spomenutých tabuľkách je možné vidieť, že vlastné implementácie medziprocesovej komunikácie pomocou FIFO zoznamov v zdieľanej pamäti sú z pohľadu testovanej situácie najlepšie. Nimi dosiahnutá výkonnosť je asi dva až trikrát vyššia ako pri použití najvýkonnejšieho zo štandardných prístupov ponúkaných Unixom, ktorým sú pomenované rúry. Výsledky taktiež ukazujú výrazne klesajúcu výkonnosť s rastúcim počtom producentov pri využití spoločného komunikačného prostriedku (rúra alebo front správ) s konzumentom pre všetkých z nich. Všeobecne je lepšie vytvoriť vlastný komunikačný prostriedok s konzumentom pre každého producenta. Pozorovať je možné aj nepatrné zhoršenie výkonnosti všetkých testovaných spôsobov komunikácie pri použití medzi vláknami oproti použitiu medzi procesmi.

6.2 Knižnica libSDM

Knižnica implementuje detaily komunikácie medzi užívateľskými aplikáciami a SDM radičom. Poskytuje tak softvérovým aplikáciám istú formu rozhrania na priamy prístup ku kontrole fungovania SDM systému a riadeniu jeho akceleračnej časti. V rámci knižnice je definovaný formát identifikátoru toku použitého v SDM, formát pravidla o toku aj s funkciami na prácu s ním a samozrejme funkcie na samotnú komunikáciu s radičom – nastavovanie platných kontextov, registráciu a rušenie prístupových kanálov a posielanie pravidiel o tokoch prístupovými kanálmi. Implementácia knižnice libSDM je realizovaná v jazyku C a jej podrobný popis je možné nájsť v anglickej implementačnej dokumentácii vytvoriteľnej programom `doxygen` (viď príložené CD).

Identifikátor toku (štruktúra `sdm_flow_identifier_t`) je v SDM založený na štandardne používanej pätici – zdrojová a cieľová IP adresa, zdrojové a cieľové číslo portu a číslo protokolu transportnej vrstvy. Keďže SDM je vytvorený s ohľadom na použitie v IPv4 aj IPv6 sieťach je uvedená pätica doplnená ešte o číslo použitej verzie IP protokolu. Vďaka tomu je možné jednoznačne rozlíšiť ako pracovať s IP adresami v identifikátore, či ako s IPv4 alebo IPv6. Okrem identifikátora konkrétneho toku definuje knižnica libSDM aj identifikátor skupiny tokov (štruktúra `sdm_flow_group_identifier_t`). Ten doplňuje základný identifikátor toku o možnosť definovať niektoré jeho položky ako nezaujímavé a navyše dovoľuje použitie prefixov zvolenej dĺžky namiesto celých IP adries. Identifikátory skupiny tokov predstavujú základ pre definovanie kontextov (štruktúra `context_t`). Každý kontext okrem množiny identifikátorov skupín tokov, ktoré sú preň zaujímavé, obsahuje aj nastavenie implicitnej akcie použitej na predspracovanie nových zaujímavých tokov a rozsah DMA kanálov, na ktoré majú byť dáta zaujímavých tokov smerované z hardvéru.

Formát pravidla o toku resp. správy posielanej SDM radiču (štruktúra `sdm_message_t`) obsahuje identifikátor konkrétneho toku doplnený o ku nemu patriacu informáciu. Tá je zložená z operácií, akcie a kontextov. Operácia definuje ako sa má správa vyhodnotiť v SDM – zmazať záznam o toku ak existuje, označiť tok za ťažký, nastaviť nový spôsob požadovaného predspracovania toku v daných kontextoch alebo bez ohľadu na kontexty. Akcia má zmysel len pri nastavovaní nového predspracovania toku a definuje jeho konkrétny požadovaný spôsob – celé rámce, UH hlavičky, záznam o toku alebo zahadzovanie. Nakoniec v prípade nastavenia nového predspracovania s ohľadom na kontexty je možné definovať masku kontextov, ktoré majú byť ovplyvnené. Na prácu s jednotlivými položkami pravidla (správy) sú v knižnici vytvorené základné funkcie. Týka sa to hlavne funkcií na nastavenie akcie, nastavenie operácie, prácu s maskou kontextov a prístup ku identifikátoru toku.

Na komunikáciu s radičom SDM sú v knižnici libSDM vytvorené nasledujúce funkcie prístupné užívateľovi:

Inicializácia SDM komunikácie: (`sdm_init`) umožňuje základné spojenie s radičom cez zdieľané všeobecné komunikačné rozhranie, ktoré mapuje do logického adresného priestoru aplikácie. Pred akoukoľvek ďalšou akciou so SDM je nutné zavolať túto funkciu.

Ukončenie SDM komunikácie: (`sdm_deinit`) ukončuje základné spojenie s radičom SDM, odpojením zdieľaného všeobecného rozhrania z adresného priestoru aplikácie. Je potrebné ju zavolať po vykonaní poslednej akcie so SDM a pred ukončením programu.

Vytvorenie kontextu: (`sdm_start_context`) umožňuje pridanie nového kontextu spracovania dát definovaného aplikáciou. Výsledkom funkcie je číslo priradené vytvorenému kontextu.

Zrušenie kontextu: (`sdm_end_context`) odstráni kontext spracovania dát s požadovaným číslom ak existuje.

Otvorenie komunikačného kanála: (`sdm_open`) vytvorí nový komunikačný kanál na zasielanie pravidiel do SDM radiča. Aplikácia môže mať aj viacero rôznych komunikačných kanálov.

Zatvorenie komunikačného kanála: (`sdm_close`) zruší komunikačný kanál na zasielanie pravidiel do SDM radiča. Aplikácia musí pred skončením zavrieť všetky svoje otvorené komunikačné kanály.

Zaslanie pravidla: (`sdm_send_message`) umožňuje posielať SDM radiču pravidlá (správy) popísané podrobne vyššie. Správu je potrebné zostaviť v lokálnej štruktúre a odovzdať funkcii odkaz na ňu, aby ju mohla skopírovať do pamäte komunikačného kanála a odoslať.

Zaslanie pravidla bez kopírovania: (`sdm_get_message_buffer` nasledované `sdm_send`) je umožnené získaním prístupu priamo do pamäte komunikačného kanála určenej na ďalšie zasielané pravidlo. Vďaka tomu je možné vyplniť položky pravidla priamo na mieste a následné len potvrdiť jeho odoslania bez nutnosti kopírovania pamäte.

Komunikačné kanály medzi aplikáciami a SDM radičom sú v knižnici libSDM implementované využitím vlastnej realizácie FIFO zoznamov v zdieľanej pamäti. Bolo tak urobené na základe výsledkov testovania výkonnosti rôznych spôsobov medziprocesorovej komunikácie uvedených v predošlej sekcii. Konkrétne je použitý variant využívajúci vstavané atomické funkcie prekladača `gcc` aj napriek tomu, že dosahuje o trochu horšiu výkonnosť ako druhá predstavená možnosť realizácie. Tento variant je však bezpečnejšie použiť z dôvodu nezávislosti na detailoch konkrétnej platformy. V budúcnosti je možné realizáciu FIFO zoznamov pretvoriť s využitím štandardných funkcií jazyka C na atomický prístup k pamäti definovaných v najnovšom štandarde jazyka C (C11)[31]. V súčasnosti ich však prekladače ešte nepodporujú.

Na demonštráciu základného použitia knižnice libSDM sú spolu s ňou implementované dva jednoduché programy – `context_editor` a `access_tester`. Prvý z nich implementuje operácie potrebné na správu kontextov. Je schopný vytvoriť a aktivovať nový kontext na základe jeho textovej definície prečítanej zo súboru a na výstupe zobraziť jemu pridelené číslo. Zároveň vie zrušiť existujúci kontext na základe jeho čísla. Druhý z demonštračných programov implementuje funkcionalitu potrebnú na posielanie pravidiel o predspracovaní tokov SDM radiču. Jeho priebeh teda začína vytvorením komunikačného kanálu so SDM radičom, pokračuje poslaním pravidiel definovaných v jeho parametroch a končí správnym zavretím komunikačného kanálu.

6.3 Softvérový radič SDM

Radič SDM implementuje užívateľskými aplikáciami riadenú konfiguráciu firmvérovej časti SDM systému. Predstavuje centralizované jadro softvérového ovládania hardvérovej akcelerácie predspracovania sieťových dát. Vytvorená implementácia sa pridŕža podrobného návrhu softvéru SDM uvedeného v sekcii 5.2. Využíva pri tom niektoré časti knižnice libSDM pre zabezpečenie komunikácie s aplikáciami. Časť radiča určená na samotnú komunikáciu s firmvérom je zatiaľ vytvorená len pre použitie so simulačným modelom popísaným v nasledujúcej sekcii. Použitie radiča nad reálnym firmvérom SDM však bude vyžadovať len

reimplementáciu funkcií priameho prístupu k nemu, veľká časť vytvoreného kódu bude použiteľná bez zmeny. Implementácia radiča SDM je realizovaná v jazyku C a jej podrobný popis je možné nájsť v anglickej implementačnej dokumentácii vytvoriteľnej programom `doxygen` (viď príložené CD).

V návrhu zo sekcie 5.2 je uvedené, že tabuľku tokov predstavujúcu základnú dátovú štruktúru SDM radiča je možné realizovať viacerými spôsobmi. Vo vytvorenej implementácii bol zvolený variant využitia haš funkcie, konkrétne je použitá MD5 haš z OpenSSL knižnice. Tabuľku tokov je tak možné označiť za haš tabuľku. Prístupový index aj index voľných záznamov je implicitne realizovaný samotnou haš funkciou a nie je potrebné ich špeciálne ukladať. Veľkou výhodou požitia haš tabuľky tokov je vyhľadanie a prístup k položkám podľa identifikátoru toku (definovaného v libSDM) s malým počtom pamäťových prístupov, ktoré sú navyše navzájom nezávislé. Nevýhodou je nevyužiteľnosť plnej kapacity tabuľky z dôvodu kolízií hašov záznamov. Na zníženie negatívneho efektu kolízii na dosiahnuteľné zaplnenie tabuľky je vhodné využívať istý stupeň asociatívnosti položiek tabuľky. Haš identifikátoru tak neukazuje na polohu jediného záznamu v haš tabuľke, ale radšej na skupinu niekoľkých rôznych záznamov, z ktorých ľubovoľný môže byť použitý.

Návrh zo sekcie 5.2 ďalej nerieši ani presné podrobnosti vhodnej implementácie cyklického získavania príkazov z rôznych zdrojov na vykonávanie v softvérovom vlákne resp. hardvérovom. Cyklické testovanie môže naznačovať využitie jednoduchého riešenia pomocou aktívneho čakania neustálym overovaním zdrojov vstupných udalostí. Uvedené jednoduché riešenie je výhodné v prípade neustálej relatívne vysokej úrovne vyťaženia vstupov, ale v prípade nízkeho vyťaženia vedie k zbytočnej spotrebe procesorového času neustálym nepotrebným testovaním príchodu udalostí. Vhodné je preto navrhnúť pokročilejšie riešenie. Vo vytvorenej implementácii SDM radiča je na uvedenú úlohu použitý princíp cyklického testovania vstupov obohatení o použitie pomocného semaforu na umožnenie pasívneho čakania vo vhodných okamihoch. V prípade neustáleho vyťaženia vstupných zdrojov udalostí sú tieto zdroje cyklicky testované bez použitia semaforu (práca so semaforom je drahá operácia). Až v prípade poklesu frekvencie príchodu udalostí tak, že sú v jednom cykle všetky prejdené zdroje udalostí prázdne, je nastavený príznak na využitie semaforu. Navyše je ale spravený ešte jeden cyklus overenia prázdnosti všetkých zdrojov pre prípad, že by vstupná udalosť prišla práve medzi testovaním vstupov a nastavením použitia semaforu. V prípade, že ani druhý cyklus neobjaví vstupnú udalosť, je proces uspaný pri snahe zamknúť semafor. Zdroje vstupných udalostí v prípade nastaveného príznaku použitia semaforu v momente vytvorenia udalosti navyše odomykajú semafor a rušia príznak jeho použitia. Teda príchod prvej udalosti po uspaní softvérového resp. hardvérového vlákna spôsobí jeho zobudenie a zrušenie použitia semaforu do ďalšej detekcie poklesu frekvencie príchodu udalostí.

Detaily fungovania vytvoreného SDM radiča je možné nastavovať pri jeho spúšťaní pomocou parametrov príkazového riadku. Podporované je ovládanie nasledujúcich vlastností:

Stupeň asociatívnosti haš tabuľky: (`-a`) je nastavovaný ako počet spodných bitov a haš hodnoty, ktoré majú byť ignorované. Teda každý haš ukazuje na blok 2^a po sebe idúcich záznamov. Minimálna dovolená hodnota je 0, maximálna je rovnaká ako použitá veľkosť tabuľky tokov (parameter `-s`).

Automaticky ťažké toky: (`-H`) vedie k automatickému označovaniu každého pridávaného záznamu toku ako ťažký tok. Nahranie každého pravidla o toku do firmvéru je podmienené jeho označením za ťažký tok.

Maximálny počet zdrojov pravidiel: (`-i`) ktoré môžu byť súbežne registrované a aktívne. Minimálna dovolená hodnota je 0, maximálna až 250.

Veľkosť FIFO medzi vláknami radiča: (`-q`) sa nastavuje ako dvojkový logaritmus počtu jeho položiek. Minimálna povolená hodnota je 0, maximálna 31.

Veľkosť tabuľky tokov: (`-s`) sa nastavuje ako dvojkový logaritmus kapacity tabuľky tokov v počte záznamov. Minimálna povolená hodnota je 0, maximálna 31.

Časové intervaly: (`-t`) sa nastavujú na počet celých sekúnd. Nastaviteľná je doba životnosti pravidiel o tokoch aj časový interval pravidelného exportu záznamov z firmvéru.

Presnosti merania intervalov: (`-T`) definujú presnosť merania dĺžky oboch časových intervalov a nastavujú sa v sekundách. Použijú sa na samotné nastavenie intervalu oboch časovačov použitých v SDM radiči na pravideľný posun o blok v im zodpovedajúcom časovom indexe.

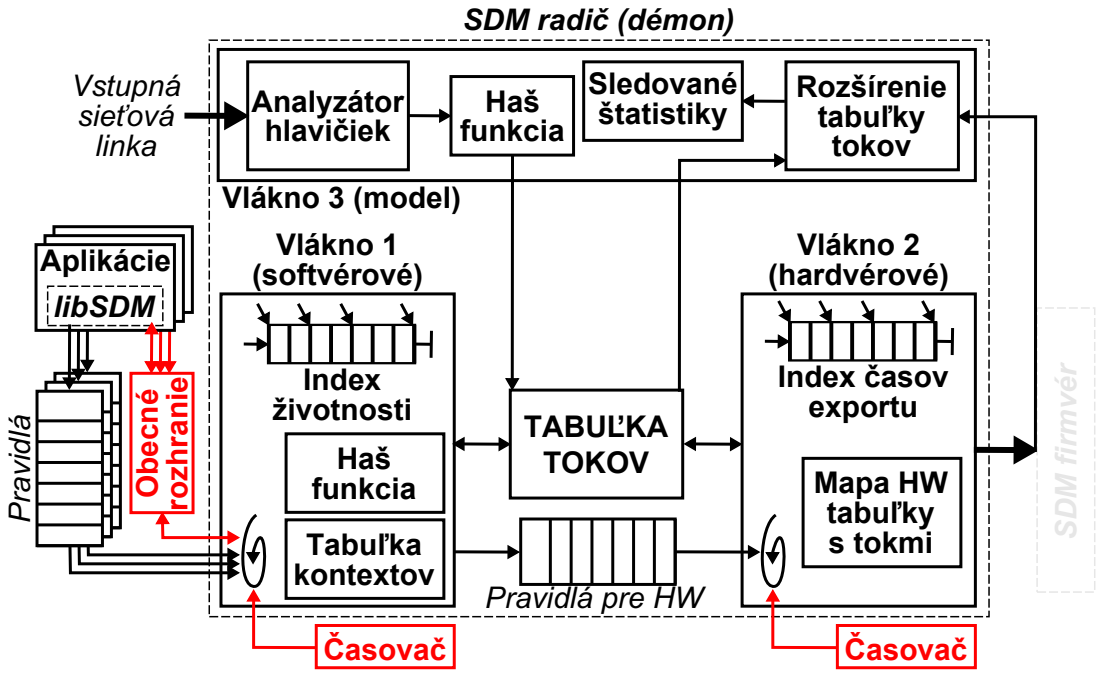
Použité implicitné hodnoty jednotlivých parametrov použité pri ich neuvedení je možné zistiť pomocou parametru `-d`.

6.4 Simulačný model

Hlavným dôvodom vytvorenia simulačného modelu je umožnenie merania dosiahnuteľných parametrov predspracovania sieťového toku dát využitím SDM systému pri nasadení s rôznymi monitorovacími a bezpečnostnými aplikáciami. Model teda bude musieť byť schopný istým spôsobom simulovať a sledovať spôsob predspracovania dátového toku zo siete podľa požiadaviek od aplikácií. Príjem a spracovanie aplikačných požiadaviek vie do veľkej miery riešiť už navrhnutý a implementovaný softvérový radič SDM. Simulačný model je tak možné vytvoriť s jeho použitím a rozšírením. Konkrétne stačí presmerovať konfiguračnú cestu z radiča určenú pre firmvér, aby smerovala do vytváraného simulačného modelu. Samotná funkcionalita simulačného modelu tak bude spočívať v sledovaní rôznych štatistík o predspracovaní vstupného sieťového toku podľa pravidiel prijímaných od SDM radiča. Popísaný postup vytvorenia modelu systému poskytuje zároveň presnejšie výsledky dosahované simuláciou, pretože softvérová časť SDM je použitá v podstate bez zmeny a simulovaná je len funkcionalita firmvérovej časti systému. Taktiež aplikácie upravené pre potreby použitia pri simuláciách SDM nie je potrebné následne upravovať pri použití s reálnym systémom.

Schéma implementovaného softvérového radiča SDM obohateného o model firmvéru je znázornená na obrázku 6.1. Na schéme v porovnaní so základným konceptom radiča z obrázku 5.2 vidno aj úpravu použitia indexov tabuľky tokov podľa vytvorenej implementácie. Konkrétne ide o nahradenie prístupového indexu a indexu voľných položiek za haš funkciu. Hlavnou zmenou v schéme je však pridanie vlákna modelujúceho firmvér. Jeho vstupmi sú, ako už bolo vyššie spomínané, dáta z rovnakej sieťovej linky akú sledujú aplikácie (zľava) a pravidlá určené pre firmvér od hardvérového vlákna SDM firmvéru (sprava). Modelové vlákno má okrem toho prístup k hlavnej tabuľke tokov (zdola) pomocou rovnakej haš funkcie akú má aj softvérové vlákno. Model pristupuje k tabuľke tokov len kvôli vyhľadávaniu a čítaniu identifikátorov tokov, dáta v tabuľke nijako neupravuje.

Samotnú funkcionalitu vlákna modelujúceho firmvér SDM je možné rozdeliť na štyri základné časti:



Obr. 6.1: Schéma softvérového radiča SDM doplneného o model firmvéru

Analýza hlavičiek rámcov: s cieľom extrahovať položky tvoriace identifikátor toku. Z rámcov je získavaná rovnaká šestica akú definuje a používa knižnica libSDM na identifikáciu tokov. Samotný implementovaný analyzátor podporuje spracovanie nasledujúcich hlavičiek: Ethernet, VLAN, MPLS, IPv4, IPv6 (vrátane rozširujúcich), UDP a TCP.

Výpočet haš funkcie: z identifikátora toku získaného z rámcov rovnakým spôsobom ako používa softvérové vlákno SDM radiča. Potrebné je všetkým prichádzajúcim rámcom zo siete nájsť adresu záznamu o toku, ak existuje.

Rozšírenie tabuľky tokov: umožňuje modelu ukladať dodatočné informácie k jednotlivým tokom. Ukladaný je ich aktuálny spôsob predspracovania nastavený hardvérovým vláknom radiča a prípadne istá stavová informácia o nich (napr. príznak príchodu rámca patriaceho danému pravidlu od jeho vytvorenia). Adresovanie rozšírenia je rovnaké ako základnej tabuľky tokov. Na základe vyhľadania identifikátoru rámca v základnej tabuľke tokov je tak možné jednoznačne určiť jemu patriacu doplňujúcu informáciu v modeli.

Sledované štatistiky: zahŕňajú v sebe všetky informácie sledované pri simulácii modelu. Ide najmä o rôzne parametre simulovaného predspracovania sieťových dát a prijímaných konfiguračných príkazov od SDM radiča. Implementované je napríklad počítanie paketov a bajtov pre jednotlivé typy požadovaného predspracovania, počet konfiguračných príkazov od radiča, maximálny počet súčasne aktívnych pravidiel o tokoch alebo počet vyvolaných exportov záznamov o tokoch.

Bližší náhľad na fungovanie simulačného modelu SDM firmvéru je možné získať aj z popisu základných operácií, ktoré realizuje. Jedná sa v podstate len o dve základné operácie: spracovanie rámca a vykonanie príkazu od SDM radiča. Ich podrobnejší popis je uvedený v nasledujúcom texte sekcie.

Spracovanie rámca. Začína príjmom rámca na vstupnom sieťovom rozhraní a analyzovaním jeho hlavičiek. Z nich sú extrahované položky tvoriace identifikátor toku. Z identifikátoru je vypočítaný haš a použitý pri vyhľadaní záznamu o toku v hlavnej tabuľke tokov SDM radiča. Výsledkom vyhľadania je adresa záznamu o hľadanom toku v tabuľke tokov alebo informácia o jeho nenájdení. Adresa je následne využitá pri prístupe k dodatočným informáciám o tokoch uloženým v modeli, ktoré určia požadovaný spôsob predspracovania pre rámec. Na základe spôsobu predspracovania a vlastností rámca sú nakoniec zodpovedajúcim spôsobom upravené sledované štatistiky. Popísané spracovanie môže bežať aj paralelne pre viacero rámcov, implementovaný model firmvéru totiž podporuje rozdelenie spracovania dát sieťovej linky medzi viacero súbežných vlákien.

Vykonanie príkazu SDM radiča. Pri použití modelu firmvéru konfiguruje hardvérové vlákno SDM radiča modelové vlákno namiesto reálneho firmvéru. Realizuje to volaním funkcií priamo ovplyvňujúcich informácie uložené v rozšírení tabuľky tokov uloženej v modelovom vlákne. Hardvérové vlákno vždy pri konfigurácii firmvéru pozná identifikátor toku a adresu jeho záznamu v tabuľke tokov. Pri práci s reálnym firmvérom musí používať celý identifikátor toku, pretože firmvér nemá prístup do softvérovej tabuľky tokov. Pri práci s modelom firmvéru je však jednoduchšie použiť len adresu do tabuľky tokov, ktorá je priamo aj adresou do rozšírenia tabuľky tokov. Realizácia jednotlivých príkazov od SDM radiča môže okrem zmien v rozšírení tabuľky tokov viesť aj k istej úpravu sledovaných štatistík.

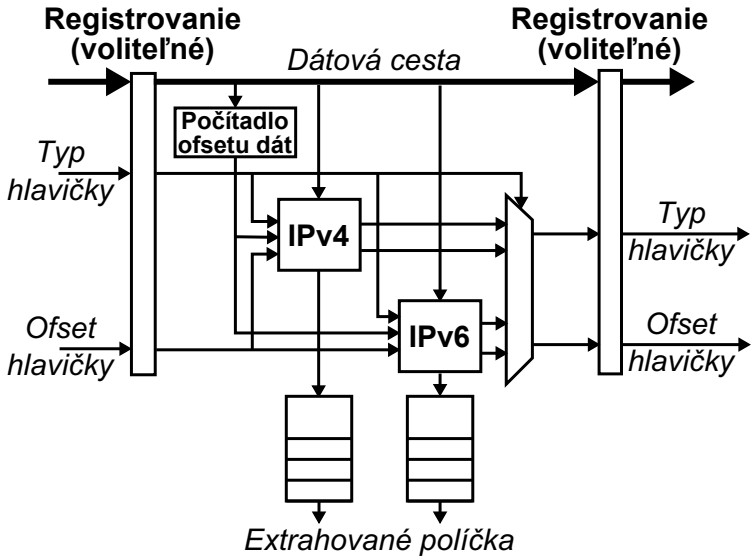
6.5 Analyzátor hlavičiek paketov

Analýza rámcov (paketov) patrí medzi základné operácie vykonávané na všetkých miestach sieťovej infraštruktúry. Moderné vysokorýchlostné siete majú zároveň vysoké požiadavky na výkonnosť a flexibilitu vytváraných analyzátorov. Tie často dosahujú potrebnú výkonnosť len za cenu vysokej spotreby zdrojov FPGA čipu. Vysoká spotreba je vo väčšiny prístupov spôsobená ich slabou škálovateľnosťou vzhľadom na rastúcu šírku použitej dátovej cesty potrebnú pre dosiahnutie vysokej priepustnosti. Niektoré riešenia zase vysokú priepustnosť získavajú zvyšovaním pracovnej frekvencie, ktoré dosahujú nadmerným zreťazením a preregistrovaním spracovania. Tento prístup okrem vysokej spotreby zdrojov naviac zvyšuje latenciu spracovania.

Analyzátor hlavičiek rámcov vytvorený nad rámec tejto diplomovej práce pre použitie v SDM firmvéri sa vyššie uvedeným problémom snaží vyhnúť. Využíva na to novú inovatívnu architektúru zreťazeného spracovania, ktorá mu umožňuje okrem dosahovania vysokej priepustnosti (nad 100 Gb/s) spracovanie rámcov s nízkou latenciou. Navyše latencia, priepustnosť a spotreba zdrojov môžu byť veľmi presne prispôsobené pre potreby konkrétnej aplikácie. Vytvorený analyzátor je ručne optimalizovaný vďaka jeho priamej implementácii v jazyku VHDL (bez prostriedkov HLS), zároveň je ale jeho štruktúra veľmi uniformná a jednoducho rozšíriteľná o podporu spracovania nových protokolov.

Návrh architektúry analyzátora začal voľbou vhodného spôsobu realizácie dátovej cesty na prenos rámcov. Keďže kvôli potrebe vysokej priepustnosti je očakávané použitie veľkých dátových šírok, je potrebné zamerať sa na riešenie problému efektivity prenosu. Problém efektívneho prenosu rámcov na vysokých dátových šírkach je už raz popísaný a riešený v sekcii 5.1 tejto práce. V návrhu a implementácii analyzátora je použité rovnaké riešenie ako je navrhnuté v uvedenej sekcii.

Ďalším krokom návrhu je realizácia jednotlivých modulov na spracovanie hlavičiek rôznych sieťových protokolov. Kvôli jednotnosti ich vzájomného zapojenia a tvorby je pre ne

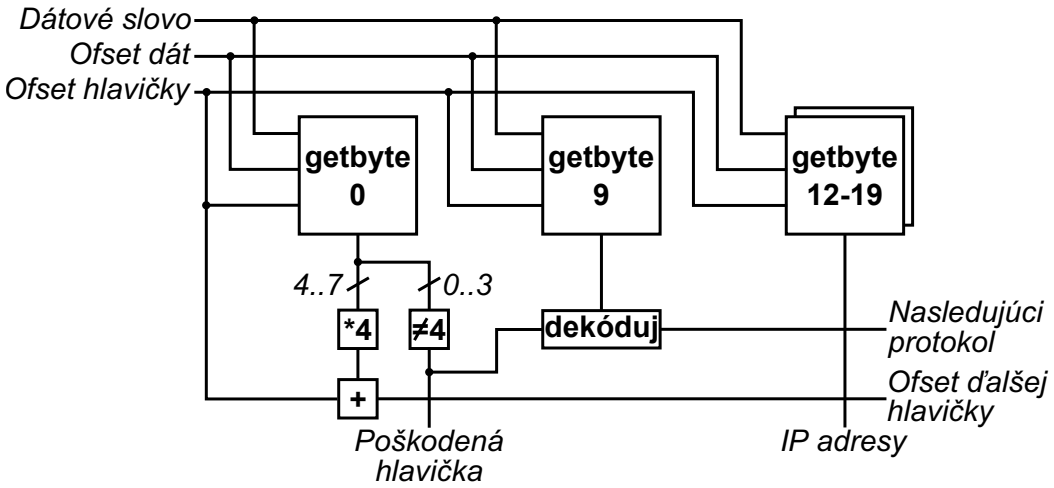


Obr. 6.2: Príklad jedného kroku zreťazeného analyzátora

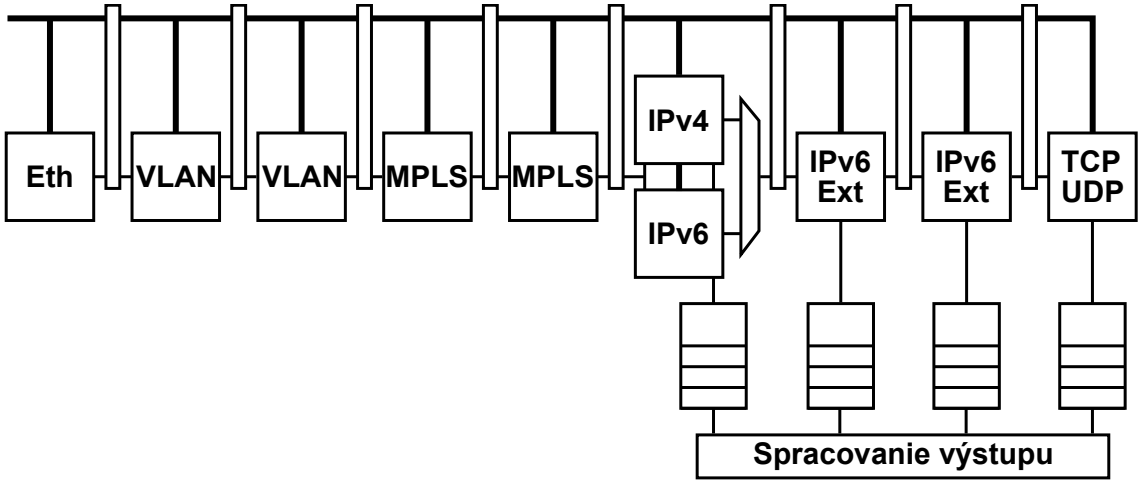
vytvorené jednotné rozhranie označené GPPI (Generic Protocol Parser Interface). Toto rozhranie obsahuje potrebné vstupné informácie na spracovanie jednej hlavičky protokolu: aktuálne dátové slovo rámca, ofset začiatku prenášaného slova a ofset začiatku hlavičky. Výstupné informácie GPPI zahŕňajú extrahované položky spracovanej hlavičky a informácie potrebné na analýzu nasledujúcej hlavičky: nasledujúci protokol a ofset začiatku samotnej hlavičky. Príklad zapojenia analyzačných modulov a použitia GPPI je možné vidieť na obrázku 6.2. Naľavo blokov IPv4 a IPv6 je vidieť popísanú vstupnú časť GPPI, vpravo a dole popísanú výstupnú časť. Obrázok tiež zobrazuje nastaviteľné preregistrovanie zreťazeného spracovania. Dodržaním jednotného rozhrania pri implementácii modulov je dosiahnutý náznak objektovej orientácie a dedičnosti vo VHDL.

Napriek tomu, že vnútorná implementácia analyzačných modulov jednotlivých protokolov je vzájomne rozdielna, je možné medzi nimi nájsť isté spoločné znaky. Základom analýzy skoro všetkých hlavičiek je čakanie na výskyt jednotlivých bajtov istej jej položky v aktuálnom slove dátovej cesty. Inak povedané, čakanie na splnenie podmienky $p + f \in \langle d; d + w \rangle$, kde p je ofset hlavičky protokolu (zo vstupu), f je ofset daného bajtu položky od začiatku hlavičky (podľa špecifikácie protokolu), d je ofset aktuálneho dátového slova (zo vstupu) a w je šírka jedného dátového slova. Pri rozpoznaní požadovaného bajtu v dátovom slove je jeho hodnota odčítaná a uložená pre ďalšie využitie, akým môže byť napríklad rozpoznanie nasledujúceho protokolu alebo určenie dĺžky skúmanej hlavičky. Práve popísanú funkcionalitu spojenú s odčítaním konkrétneho bajtu z dátovej cesty realizuje jednotka označená `getbyte`, na ktorej využití je možné jednoducho postaviť implementáciu modulov na spracovanie skoro všetkých protokolov. Príklad analyzátora hlavičky IPv4 protokolu je ilustrovaný na obrázku 6.3. Zo schémy na obrázku je jasne viditeľné, že okrem použitia `getbyte` jednotiek postačuje implementovať už len minimum ďalšej špecifickej funkcionality okolo. Konkrétne v tomto prípade: výpočet dĺžky a overenie správnosti IPv4 hlavičky (z bajtu 0), dekódovanie nasledujúceho protokolu (z bajtu 9).

Po vytvorení potrebných analyzačných modulov je možné prejsť k tvorbe samotnej architektúry analyzátora. Základom pri tom je zreťazenie spracovania umožňujúce dosiahnutie vysokej priepustnosti. Avšak dĺžka každého kroku zreťazenia je nastaviteľná pomocou



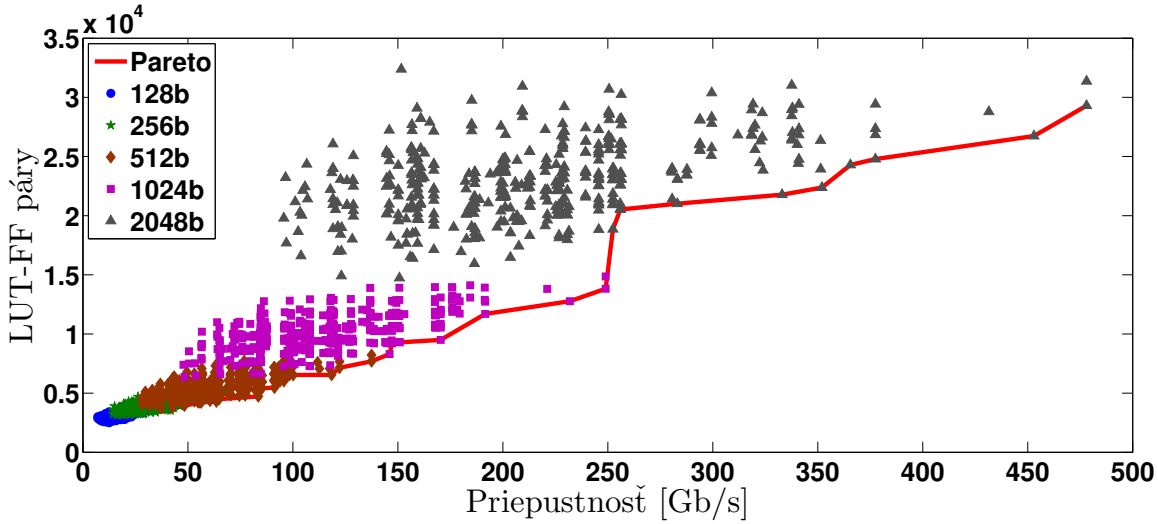
Obr. 6.3: Príklad použitia `getbyte` v IPv4 analyzačnom module



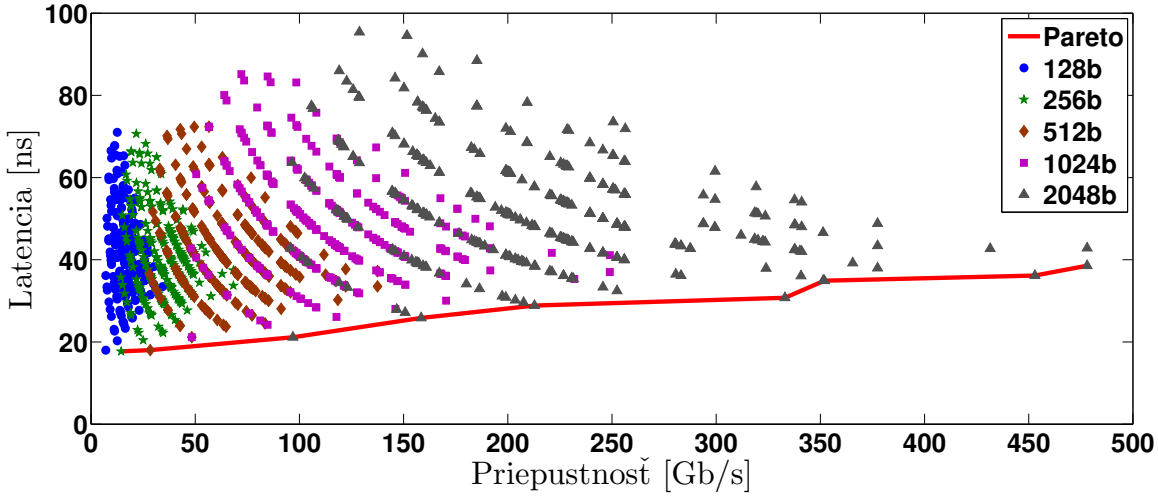
Obr. 6.4: Celková schéma analyzátora hlavičiek vytvoreného pre potreby SDM

voliteľných preregistrovaní vkladaných medzi jednotlivé kroky analýzy (ako vidno aj na obrázku 6.2). Zreťazenie s malým krokom (veľa povolených registrov) zvýši pracovnú frekvenciu a priepustnosť, ale zároveň aj latenciu a využité zdroje FPGA. Povoľovaním rôznych registrovaní je tak možné ladiť parametre analyzátora podľa potrieb konkrétneho použitia.

Každý analyzačný modul obsahuje oproti obrázku 6.3 ešte navyše vnútorné preskočenie spracovania hlavičky v prípade, že patrí protokolu, ktorý nie je modulom podporovaný. Preskočenie spracovania spočíva v posunutí vstupných údajov GPPI na výstupné bez zásahu do nich. Vďaka podpore popísanej funkcionality je možné celkovú architektúru analyzátora vytvoriť ako jednoduchú linku za sebou idúcich analyzátorov jednotlivých protokolov. Príklad celkovej architektúry analyzátora je zobrazený na obrázku 6.4. Jedná sa o architektúru analyzátora poskladanú pre využitie v SDM firmvéri. Všimnúť si je možné popísané jednoduché lineárne zapojenie modulov analyzujúcich jednotlivé protokoly. Vďaka podpore preskočenia spracovania hlavičky istým modulom je navyše možné zobrazeným analyzátorom úspešne spracúvať aj rámce s nižším počtom VLAN, MPLS a IPv6 rozširujúcich hlavičiek ako je zobrazené.



Obr. 6.5: Závislosť spotreby zdrojov FPGA od priepustnosti pre rôzne konfigurácie



Obr. 6.6: Závislosť dosahovanej latencie od priepustnosti pre rôzne konfigurácie

Analyzátor z obrázku 6.4 navrhnutý pre potreby SDM firmvéru je kompletne implementovaný vo VHDL, verifikovaný použitím verifikačného prostredia v SystemVerilog a testovaný priamo na FPGA čipe rodiny Virtex-6. Nastavený je tak aby extrahoval položky zaujímavé pre SDM firmvér z IP, UDP a TCP hlavičiek rámcov. Výkonnosť dosahovanú analyzátorom je možné ladiť použitým delením zreťazenia na kroky (8 miest s voliteľným preregistrovaním) a nastavenou šírkou dátovej cesty. Oba parametre môžu ovplyvniť dosiahnutú priepustnosť, využité zdroje aj latenciu analyzátora. Dosiahnuteľné parametre analyzátora vytvoreného pre SDM firmvér sú zanesené v grafoch na obrázkoch 6.5 a 6.6. Hodnoty v grafoch sú získané z výsledkov syntézy jednotlivých možných nastavení analyzátora pre FPGA čip Virtex-7 870HT. Prvý z grafov zobrazuje závislosť využitých zdrojov čipu na dosiahnutej priepustnosti. Druhý zobrazuje závislosť latencie na priepustnosti. V grafoch je každé konkrétne nastavenia analyzátora zanesené ako jeden bod. Nastavenia s rovnakou použitou šírkou dátovej cesty sú označené rovnakou farbou aj symbolom. V grafoch je navyše červenou čiarou vyznačená množina pareto optimálnych konfigurácií analyzátora.

Pre porovnanie zreťazený analyzátor popísaný v [32] dosahuje pri veľmi podobnej podpore spracovania protokolov a rovnakom použitom FPGA čipe parametre: priepustnosť 325 Gb/s, spotreba 67 902 LUT-FF párov a latencia 309 ns, čo je niekoľko násobne horšie ako dosahuje vytvorený analyzátor.

Prieskumom priestoru možných riešení je možné nájsť vhodnú konfiguráciu prezentovaného analyzátora hlavičiek rámcov pre potreby SDM firmvéru. Konfigurácia vybraná ako najlepšia používa 512 bitov širokú dátovú cestu a je schopná pracovať na frekvencii nad 200 MHz s dosahovanými parametrami: priepustnosť nad 100 Gb/s, spotreba 6 536 LUT-FF párov a latencia 35,8 ns. Použitie vytvoreného analyzátora hlavičiek je vďaka jeho flexibilite, jednoduchej rozšíriteľnosti a dosahovanej výkonnosti výrazne širšie ako len pre potreby SDM. Vysoká kvalita analyzátora bola ocenená aj na odbornej ACM/IEEE konferencii ANCS v Austine zameranej na sieťové a komunikačné systémy, kde bol jeho návrh prezentovaný [33].

6.6 Klasifikátor tokov

Klasifikácia rámcov (paketov) patrí medzi základné operácie využívané v kľúčových miestach sieťovej infraštruktúry. Moderné vysokorýchlostné siete majú vysoké požiadavky na priepustnosť a kapacitu pravidiel klasifikácie. Najrozšírenejšia je klasifikácia podľa IP adries alebo prefixov IP adries (napr. smerovanie), aj keď často sa možeme stretnúť aj s klasifikáciou podľa tokov (napr. OpenFlow). Všeobecne je možné klasifikovať dvomi spôsobmi – úplnou alebo čiastočnou zhodou klasifikovaného kľúča s kľúčom pravidla. Pri úplnej zhode je možné jednotlivé prvky oboch kľúčov porovnať jednoducho na rovnosť. Čiastočná zhoda ale navyše zavádza možnosť použitia špeciálnych hodnôt (napr. vždy zhodné) namiesto istých položiek kľúčov pravidiel, prípadne realizuje porovnanie prvkov na zhodu sofistikovanejším spôsobom (napr. stačí zhoda prefixu).

Klasifikátor rámcov potrebný pre SDM firmvér v sebe spája oba spôsoby porovnania. Pracuje s obmedzeným počtom relatívne statických pravidiel pre skupiny tokov, ktoré umožňujú použitie prefixov IP adries a použitie špeciálnych vždy zhodných hodnôt namiesto položiek. Vďaka nízkemu počtu a malej frekvencii zmien je možné túto časť klasifikácie riešiť použitím malej asociatívnej pamäte. Zároveň však SDM klasifikátor pracuje s veľkým počtom dynamicky sa meniacich pravidiel pre konkrétne toky. Riešenie tejto časti je kvôli potrebnej kapacite a vysokej frekvencii zmien zložitejšie a vyžaduje návrh a vytvorenie vhodnej architektúry.

Nad rámec zadania tejto diplomovej práce je rozpracovaný návrh architektúry jednotky riešiacej vyššie popísanú klasifikáciu rámcov podľa presnej zhody identifikátoru toku. Zároveň je podľa návrhu vytvorená VHDL implementácia funkčného prototypu navrhnutej architektúry, ktorá spĺňa všetky požiadavky na použitie v SDM firmvéri okrem dostatočnej kapacity, pretože zatiaľ vie využívať len interné pamäte FPGA čipu. Problém s nízkou kapacitou je teda jednoducho riešiteľný doplnením podpory využitia externej pamäti do implementovaného prototypu.

Návrh architektúry jednotky klasifikácie začal voľbou dátovej štruktúry na efektívne ukladanie a prácu so súborom pravidiel. Zvolená dátová štruktúra musí hlavne umožňovať dostatočne vysokú priepustnosť klasifikácie pre použitie v 100 Gb/s sieťach. Prakticky to znamená začatie novej klasifikácie v každom takte pri pracovnej frekvencii aspoň 200 MHz. Ďalej musí byť umožnené atomické pridávania a odoberanie jednotlivých pravidiel bez obmedzenia klasifikácie na plnej priepustnosti. Ako vhodná štruktúra bola zvolená haš tabuľka, pretože umožňuje vyhľadanie ľubovoľnému kľúču patriacej položky s minimálnym

počtom potrebných prístupov do pamäte. Záznamy ukladané v haš tabuľke majú pre potreby klasifikácie formát dvojice – kľúč (identifikátor toku) a jemu priradená trieda (spôsob predspracovania).

Pri použití haš tabuľky sú hlavným problémom vzájomné kolízie hašov kľúčov vkladaných pravidiel, vedúce k neefektívnemu využitiu jej celkovej kapacity. Inak povedané, vloženie nového pravidla do haš tabuľky môže zlyhať aj v prípade, že sú v tabuľke stále ešte voľné miesta. Znížiť negatívny dopad kolízií je možné minimalizovaním ich výskytu vhodnou voľbou použitej haš funkcie. Aj využitím veľmi zložitej a kvalitnej haš funkcie sa však nedocieli výrazný nárast dosiahnuteľného zaplnenia tabuľky. Potenciálnejšou možnosťou na minimalizovanie negatívneho dopadu kolízií je využitie sofistikovanejšieho spôsobu práce s tabuľkou. Práca s tabuľkou v sebe zahŕňa hlavne operácie vyhľadávania a vkladania záznamov.

Pri posudzovaní rôznych spôsobov práce s haš tabuľkou kapacity t sú dôležitými parametrami hlavne počty pri vyhľadaní záznamu potrebných: prístupov do pamäte (p) a rôznych haš funkcií (h). Adresy jednotlivých prístupov označím ako $i_0, i_1 \dots i_{p-1}$ a vypočítané haš hodnoty ako $x_0, x_1 \dots x_{h-1}$. Bežne používanými a rozšírenými spôsobmi práce s haš tabuľkou sú:

Naivný: ($h = p = 1$) pristupuje k jedinej položke na adrese priamo udanej vypočítanou hodnotou haš funkcie, teda $i_0 = x_0 \mod t$.

Blokový: (h=1, p voliteľné) pracuje s istým počtom po sebe idúcich položiek, kde adresu prvej priamo udáva hodnota haš funkcie, teda $i_k = (x_0 + k) \mod t$ pre $0 \leq k < p$. Naivný spôsob tak predstavuje špeciálny prípad blokového s $p = 1$.

Dva haše: (h=2, p voliteľné) pracuje podobne ako blokový, ale namiesto prístupu k po sebe idúcim položkám (vzájomná vzdialenosť 1) využíva druhú haš funkciu, aby udávala vzdialenosti jednotlivých pristupovaných položiek, teda $i_k = (x_0 + k * x_1) \mod t$ pre $0 \leq k < p$.

Kukučie hašovanie [34]: ($h = p$, p voliteľné) prístupuje na pozície priamo dané haš funkciami, teda $i_k = x_k \mod t$ pre $0 \leq k < p$. Navyše pri pridávaní záznamov využíva na riešenie kolízií reorganizáciu položiek tabuľky popísanú ďalej. Existuje aj vo variante, kedy je prístup na pozície tabuľky striktne rozdelený medzi jednotlivé haš funkcie, teda $i_k = (x_k \mod \frac{t}{p}) + k * \frac{t}{p}$ pre $0 \leq k < p$.

Príklad priebehu procesu reorganizácie položiek haš tabuľky používaný kukučím hašovaním ($p = 2$) pri výskyte kolízie vkladania záznamu je ukázaný na obrázku 6.7. Jednotlivé zobrazené kroky ilustrujú nasledujúcu funkcionalitu:

1. Vkladaný záznam X je možné vložiť na miesta $h1$ a $h2$ dané výpočtom haš funkcií, obe sú však už zaplnené (A resp. B). Vzniká teda kolízia, ktorú sa kukučie hašovanie pokúsi riešiť zmenou usporiadania položiek tabuľky.

2. Zvolí sa obeť spomedzi záznamov obsadzujúcich pozície $h1$ a $h2$, v tomto prípade A. Obeť je následne vybratá z tabuľky a na jej miesto je vložená nová položka X. Tak vznikne situácia podobná počiatočnej – je potrebné vložiť nový záznam do tabuľky.

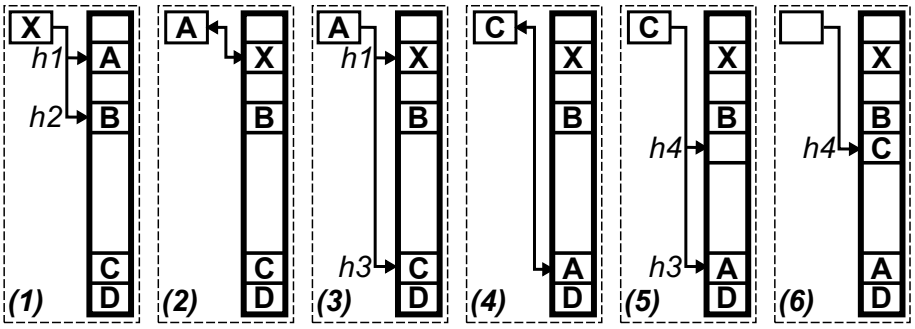
3. Podobne ako v 1., vkladaný záznam A je možné vložiť na miesta $h1$ a $h3$ dané výpočtom haš funkcií, obe sú však už zaplnené (X resp. C). Opäť je teda potrebné riešiť kolíziu zmenou usporiadania položiek tabuľky.

4. Zvolí sa obeť spomedzi záznamov obsadzujúcich pozície $h1$ a $h3$, v tomto prípade je známe, že A bolo z tabuľky práve vyňaté z pozície $h1$. Výber obete z pozície $h1$ by nás vrátil o krok späť, preto je zvolená obeť C z pozície $h3$. Všeobecne pri voľbe obete nemôže byť zvolená minulá pozícia vkladaného záznamu. Záznam C je teda vybratý z tabuľky a na jeho miesto je vložená položka A.

5. Vkladaný záznam C je možné vložiť na miesta $h3$ a $h4$ dané haš funkciami. Kolízia v tomto prípade nevznikla, lebo miesto $h4$ je prázdne.

6. Záznam C je priamo vložený na voľné miesto $h4$ a reorganizácia haš tabuľky končí úspechom.

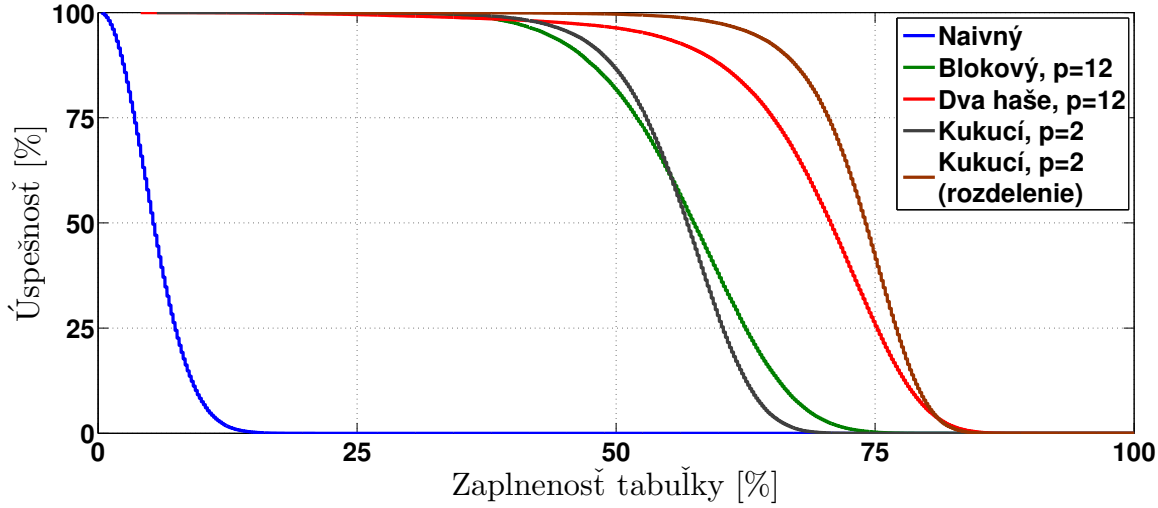
Pri reorganizácii tabuľky sa môžu kroky 3. a 4. zopakovať aj niekoľkokrát pred dosiahnutím stavu podobného kroku 5. Zároveň sa môže stať, že takýto stav nikde nenastane a reorganizácia bude prebiehať v cykle dovtedy, kým sa niektorý záznam neodstráni.



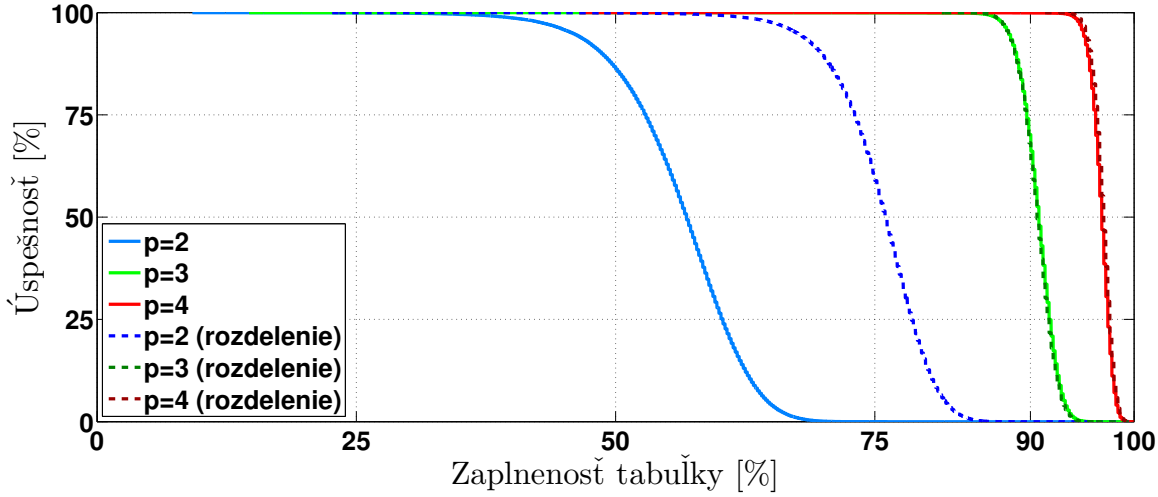
Obr. 6.7: Demonštrácia pridávania položiek pri kukučom hašovaní

Dosiahnuteľná miera zaplnenia haš tabuľky pre jednotlivé vyššie popísané spôsoby práce bola testovaná a porovnaná. Testovanie prebiehalo pre tabuľku s kapacitou $t = 8192$ položiek po 32 bitov. Použitými kľúčmi bola množina IPv4 adries extrahovaných z paketov odchytených na reálnej sieti. Pri každom teste boli z množiny adries volené náhodné položky (bez duplicít) a postupne vkladané do tabuľky až dovtedy, kým nedošlo k neúspechu vkladania. V prípade kukučieho hašovania sa za neúspešné považovalo vkladanie, ktoré vykonalo viac ako t krokov reorganizácie položiek bez nájdenia ich vhodného usporiadania. Výsledky zo 100 000 testov pre každý použitý spôsob sú vynesené v grafe na obrázku 6.8. Graf zobrazuje úspešnosť dosahovaného percentuálneho zaplnenia tabuľky. Jeden bod na grafe teda hovorí koľko percent testov daného spôsobu dosiahlo zaplnenie tabuľky aspoň daný počet percent. Graf jasne ukazuje, že naivný spôsob s dosahovaným zaplnením okolo 10 % nie je oproti ostatným riešeniam veľmi efektívny. Naproti tomu obe varianty kukučieho hašovania aj pri použití len dvoch haš funkcií a dvoch prístupov do pamäte dosahujú porovnateľné zaplnenie haš tabuľky ako blokový spôsob resp. spôsob dvoch hašov s použitím dvanástich prístupov.

Keďže sa kukučie hašovanie javí ako najlepší z testovaných spôsobov práce s haš tabuľkou je ďalej bližšie preskúmaná jeho výkonnosť pri rastúcom p. Výsledky získané testom popísaným v predchádzajúcom odseku sú zanesené v grafe na obrázku 6.9. Graf jasne ukazuje, že rastúci počet prístupov do pamäte pozitívne ovplyvňuje dosahovanú mieru zaplnenia tabuľky kukučím hašovaním. Už pri štyroch prístupoch na vyhľadanie je dosahované zaplnenie vyše 95 % kapacity tabuľky. Z grafu je zároveň viditeľné potlačenie vzájomného rozdielu kvality oboch variantov kukučieho hašovania pri p väčšom ako dva.



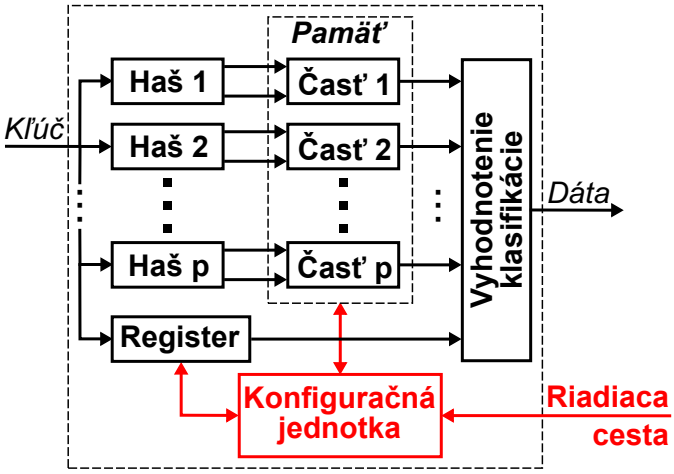
Obr. 6.8: Dosahované zaplnenia tabuľky pre rôzne spôsoby pridávania



Obr. 6.9: Dosahované zaplnenia tabuľky pre rôzne parametre kukučieho hašovania

Výsledky ukázané v grafoch jednoznačne poukazujú na vhodnosť použitia kukučieho hašovania vo vytváranej haš tabuľke. Z pohľadu hardvérovej realizácie je jednoduchší variant s implicitne rozdelenou pamäťou medzi haš funkcie, pretože v ňom nie je potrebné riešiť súbežný viacnásobný prístup do rovnakého pamäťového bloku. Po výbere haš tabuľky s kukučím hašovaním ako základ pre vytváranú klasifikačnú architektúru je potrebné už len vyriešiť, ako bude riadený priebeh reorganizácie položiek. Softvérové riešenie by viedlo k úspore zdrojov FPGA, hardvérové by zase redukovalo prácu softvéru pri pridávaní pravidiel len na definovanie hodnôt položiek pravidla a vyvolanie operácie pridania. Zvolená je možnosť riadenia realizovaného priamo v hardvéri vďaka jej výrazne nižšiemu vyťaženiu softvéru.

Návrh a vytvorenie samotnej klasifikačnej architektúry je teraz už priamočiary. Celkovú jej schému je možné vidieť na obrázku 6.10. Červenou farbou sú v ňom označené časti starajúce sa o pridávanie pravidiel a s tým súvisiacu reorganizáciu položiek. Zvyšok je potrebný na zabezpečenie samotnej klasifikácie. Architektúra je navrhnutá tak, aby podporovala ľu-



Obr. 6.10: Celková schéma haš tabuľky s využitím kukučieho hašovania

bovoľne nastaviteľný: počet použitých haš funkcií (p), veľkosť častí tabuľky v pamäti aj dátovú šírku kľúča (identifikátora) a dát (definície triedy). Bližší popis významu a funkcie jednotlivých prvkov zo zobrazenej schémy je nasledujúci:

Kľúč: jednoznačne identifikuje pravidlá v tabuľke. V prípade SDM pôjde o identifikátor toku získaný z rámca pomocou analyzátora hlavičiek popísaného v predošlej sekcii.

Dáta: informácia asociovaná s nájdeným kľúčom v tabuľke, prípadne príznak o nenájdení vhodného pravidla. V prípade SDM pôjde o definíciu spôsobu predspracovania rámcov daného toku.

Hash: jednotlivé haš funkcie určujúce adresy do častí haš tabuľky. Výpočet je implementovaný pomocou CRC bloku s predradenou konfigurovateľnou permutáciou vstupného vektoru.

Pamäť: tvorí základ pre ukladanie pravidiel tvaru kľúč a dáta. Pamäť je implicitne rozdelená na časti, kde do každej pristupuje práve jedná haš funkcia. Klasifikácia vstupného kľúča pozostáva z vyčítania záznamov z pozícií daných hašmi a porovnaním ich kľúčov so vstupným na zhodu. Informácia o nájdenej zhode a tiež dáta vybraného záznamu sú propagované ďalej.

Register: porovnanie na zhodu kľúča sa paralelne s časťami pamäte realizuje aj so záznamom v registri. Tento register je využívaný na zabezpečenie správneho fungovania hardvérom riadenej reorganizácie položiek v pamäti.

Vyhodnotenie klasifikácie: realizuje agregovanie výsledkov vyhľadania vstupného kľúča pre všetky časti pamäte aj register. Pri správnom plnení haš tabuľky, by mala obsahovať každý kľúč maximálne raz, teda každé vyhľadanie by malo skončiť nájdením maximálne jedného zhodného záznamu. Vyhodnotenie klasifikácie tak vyberá prvé dáta so záznamu s nájdenou zhodou kľúča a agreguje informáciu o úspešnosti nájdenia tejto zhody v tabuľke.

Konfiguračná jednotka: je riadená zo softvéru a realizuje všetky kroky potrebné pre atomické pridávanie a odoberanie záznamov z haš tabuľky. Prijíma len príkazy iniciujúce

vykonanie jednotlivých zmien. Jednotka využíva register na odkladanie záznamov z haš tabuľky pri jej reorganizácii. Výhodou tohto registra je jeho zapojenie do vyhodnocovacej cesty vstupných kľúčov, ktoré vedie k možnosti nepozastavovať funkčnú cestu architektúrou počas reorganizácie tabuľky. Odoberanie záznamov je realizované po jednom atomicky tak, že sa rušený kľúč postupne hľadá a prípadne ruší na každej jeho možnej pozícii (vrátane registra). Pridávanie záznamu je realizované jednoducho zápisom nového záznamu do registra. Konfiguračná jednotka následne začne reorganizáciu tabuľky (obrázok 6.7) a prehadzovaním položiek sa snaží dostať k vyprázdneniu registra. Pokým je register plný jednotka sa navonok tvári akoby mala zaplnenú haš tabuľku.

Generovanie pravidiel	**Zaplnenosť tabuľky [%]**		
	Priemer	**Minimum**	**Maximum**
Náhodne	91,9	89,7	94,0
Inkrementálne	88,6	56,6	100,0
Grayov kód	90,5	49,5	100,0

Tabuľka 6.4: Dosahovaná zaplnenosť vytvorenej haš tabuľky s využitím kukučieho hašovania

Haš tabuľka využívajúca kukučie hašovanie z obrázku 6.10 je implementovaná vo VHDL, verifikovaná použitím verifikačného prostredia v SystemVerilog a testovaná priamo na FPGA čipe rodiny Virtex-5. Pre potreby SDM je nastavená tak, aby pracovala s kľúčom predstavujúcim identifikátor toku zložený z IP adries (v4 aj v6), čísel portov a čísla protokolu. Nastaviteľný je ďalej počet a veľkosť jednotlivých častí haš tabuľky. Výsledky testov samotného hardvérového riešenia z pohľadu dosiahnutého zaplnenia pri použití 3 tabuliek po 512 položiek sú uvedené v tabuľke 6.4. Ide o výsledky z 300 testov pre každý uvedený spôsob generovania identifikátorov tokov. Generovanie prebiehalo 3 rôznymi spôsobmi:

Náhodne: každý vkladaný identifikátor je získaný úplne náhodne a nezávisle.

Inkrementálne: prvý identifikátor je náhodne vygenerovaný a nasledujúce vznikajú jeho zvyšovaním o náhodne zvolenú konštantu.

Grayov kód: prvý identifikátor je náhodne vygenerovaný a každý nasledujúci vzniká preklopením hodnoty práve jedného bitu v jeho predchodcovi.

Priemerné dosiahnuté zaplnenia hardvérovej tabuľky okolo 90 % zodpovedá očakávaným hodnotám podľa grafu 6.9. V nameraných minimálnych a maximálnych hodnotách zaplnenia je viditeľný vplyv vzájomnej podobnosti jednotlivých kľúčov vkladaných do tabuľky. Ten môže viesť k extrémne nízkemu (len okolo 50 %) alebo extrémne vysokému (až plných 100 %) dosiahnutému zaplneniu.

Pre plnohodnotné použitie v klasifikátore SDM firmvéru je potrebné zvýšiť kapacitu vytvorenej haš tabuľky implementovaním podpory uloženia jej častí aj v externej pamäti a nielen v interných pamätiach na FPGA čipe. Použitie vytvorenej implementácie haš tabuľky s kukučím hašovaním je vďaka nastaviteľným parametrom výrazne širšie ako len pre potreby SDM. Uplatnenie našla napríklad aj pri realizácii paketového filtra využívaného v hardvérovo akcelerovaných 10 Gb/s aj 1 Gb/s sondách pre systém legálneho odpočúvania vytváraný na FIT VUT v Brne v rámci projektu Sec6net [35].

Kapitola 7

Výsledky

Kapitola popisuje výsledky získané simuláciami funkcie navrhnutého SDM systému v rôznych podmienkach jeho nasadenia. Pri simuláciách sú sledované hlavne parametre dosiahnuteľnej redukcie objemu dátového toku jeho hardvérovým predspracovaním podľa aplikačných požiadaviek pri nasadení v reálnej vysokorýchlostnej sieti. Testovanie prebiehalo v sieti združenia CESNET na 10 Gb/s optickej linke medzi Brnom a Viedňou. Konkrétne išlo o linku spájajúcu siete CESNET2 a Aconet, ktorá už bola popísaná na začiatku kapitoly 4. V uvedenej kapitole sú bližšie popísané aj vlastnosti dátového toku na nej.

Parametre predspracovania sieťových dát systémom SDM je najdôležitejšie sledovať počas najväčšieho vyťaženia linky, pretože práve vtedy je najviac potrebná akcelerácia sieťových aplikácií. Simulácie systému boli preto vždy realizované práve dobe očakávaných špičiek. Linka CESNET2 – Aconet je najviac vyťažená (graf 4.1) v pracovných dňoch v čase medzi 8:00 a 16:00 kedy dátový tok dosahuje rýchlosť 3 až 4,5 Gb/s. Simulačné testy systému SDM teda prebiehali vždy v uvedených časoch a každé jedno meranie pracovalo s dátami minimálne 30 minútového intervalu. Pre väčšiu presnosť získaných výsledkov boli jednotlivé merania vždy zopakované aspoň dvakrát s istým časovým odstupom a v prípade navzájom výrazne odlišných výsledkov aj viackrát.

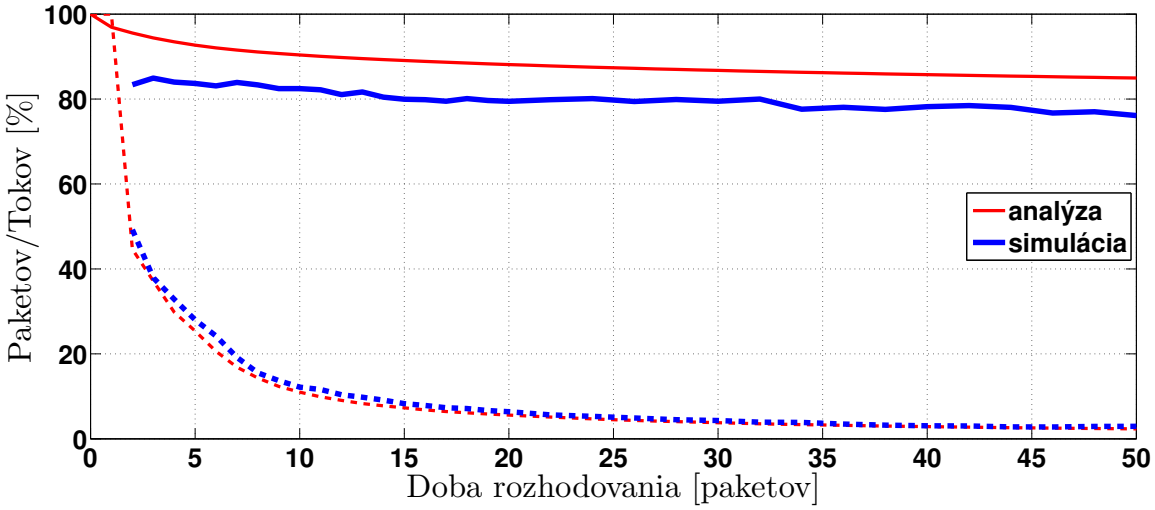
7.1 Základné vlastnosti

Simulovaný systém SDM prináša urýchlenie sieťových aplikácií plynúce hlavne zo softvérového riadenia predspracovania dát v hardvéri. Ovládať spôsob predspracovania je možné najmä pomocou aplikáciami za behu definovaných pravidiel pre jednotlivé sieťové toky. Pravidlá sú štandardne vytvárané ako reakcia na spracovanie začiatočných paketov toku a definujú spôsob predspracovania nasledujúcich paketov daného toku. V systéme teda existuje isté oneskorenie medzi vznikom pravidiel a samotným začiatkom toku, ktoré ovplyvňujú. Dĺžka uvedeného oneskorenia ovplyvňuje podiel vytváranými pravidlami ovplyvnených paketov. Základný pohľad na dosiahnuteľnú efektivitu systému SDM je tak možné získať práve skúmaním tejto jeho schopnosti.

Na testovanie schopnosti SDM ovplyvňovať spracovanie paketov na základe pravidiel o tokoch je simulovaná jednoduchá situácia, kedy softvér zaujíma len istý počet prvých paketov každého toku. Všetky pakety nových tokov sú teda implicitne posielané nezmenené do softvéru. Softvér sleduje prichádzajúce pakety a zaradzuje ich do tokov. Po prijatí daného počtu paketov istého toku rozhodne o zahadzovaní ďalších paketov tohto toku. Uvedené rozhodovanie je rovnaké ako u naivného spôsobu detekcie ťažkých tokov predstaveného

v sekcii 4.4. Pri simulácii je v systéme SDM popísané chovanie dosiahnuté vytvorením jediného kontextu s implicitným pravidlom na posielanie celých paketov do softvéru. Počas behu sú potom aplikáciou pridávané pravidlá s akciou na zahadzovanie paketov. Ako aplikácia je použitý program `flowmonexp` (popísaný už v kapitole 4) s vlastným vstupným pluginom posielajúcim pravidlá SDM radiču pomocou knižnice libSDM. Použitý plugin vznikol úpravou základného vstupného pluginu.

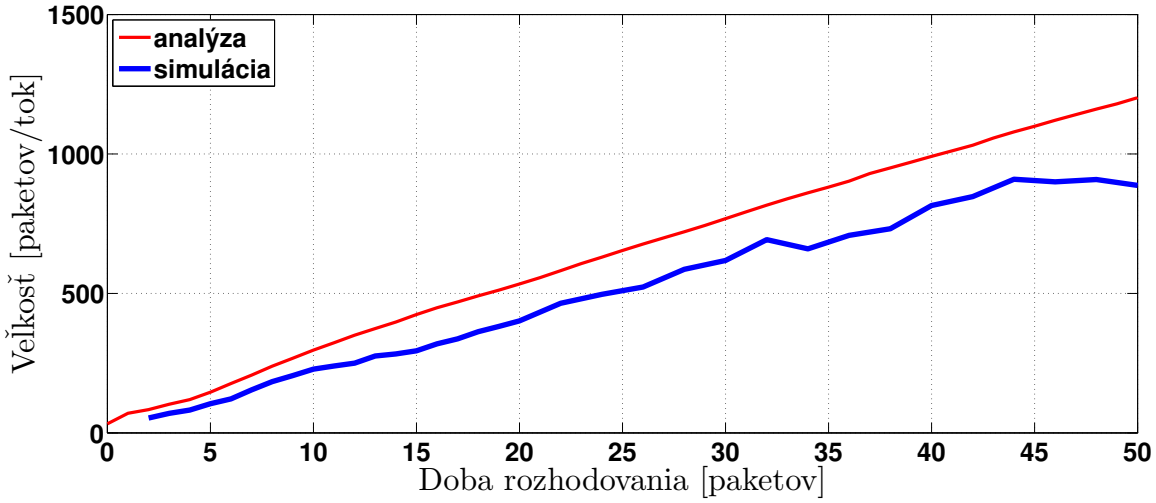
Na základe výsledkov simulácie SDM vo vyššie popísanej modelovej situácii bol vytvorených graf na obrázku 7.1. Graf ukazuje percentuálne podiely paketov resp. tokov sledovaných vlastností v závislosti na zvolenom počte paketov toku potrebných na rozhodnutie softvéru o pridaní pravidla pre tento tok. Celkový podiel paketov ovplyvnených (zahodených) na základe všetkých dynamicky pridaných pravidiel je zobrazený plnou modrou čiarou. Doplnený je aj celkový podiel tokov, pre ktoré bolo vytvorené pravidlo (prerušovaná modrá čiara). Pre porovnanie sú v grafe zanesené aj analyticky vypočítané priebehy rovnakých závislostí (červené), ktoré sú získané z grafu 4.11. Zobrazené výsledky simulácie SDM ukazujú, že dynamicky pridávané pravidlá sú schopné ovplyvniť predspracovanie až 85 % všetkých paketov a že tento podiel má mierne klesajúcu tendenciu s rastúcou dobou rozhodovania o vytvorení pravidla. Viditeľný asi 10 % rozdiel simuláciou zisteného priebehu od analytického je spôsobený zanedbaním doby trvania softvérovej spätnej väzby systému v analytickom riešení. Výsledok simulácie teda presnejšie zodpovedá parametrom reálneho správania sa SDM systému.



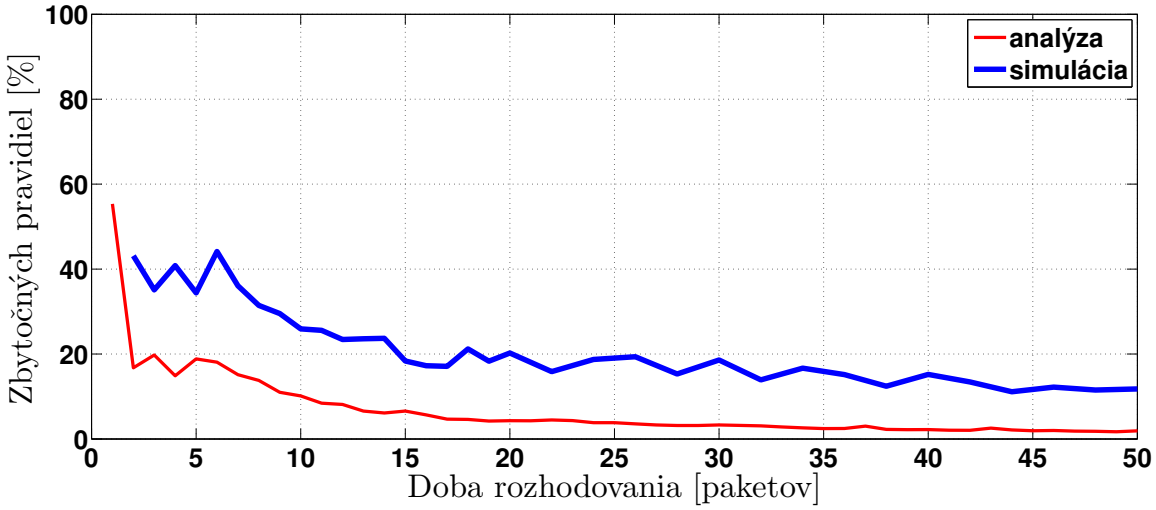
Obr. 7.1: Schopnosť SDM ovplyvňovať predspracovanie paketov v tokoch

V grafe 7.1 je ďalej pozorovateľný výrazne rýchlejší poklesu podielu tokov s pravidlom oproti nimi zachytenému podielu paketov. Teda rovnaká vlastnosť ako už bola predstavená v sekcii 4.4 v súvislosti s efektivitou detekcie ťažkých tokov jednoduchou metódou. V grafe zanesené výsledky simulácie poukazujú na efektívnu využiteľnosť tejto jednoduchej metódy aj v reálnom SDM systéme. Podrobnejší náhľad poskytuje graf 7.2. Ten ukazuje priemerný počet paketov zachytených jedným pravidlom o toku v závislosti od hraničného počtu paketov pre rozhodovanie o pridaní pravidla. Podobne ako analytické (červená podľa grafu 4.13) aj simulačné výsledky (modrá) ukazujú efektivitu jednoduchej detekcie tokov.

Efektivita pridávaných pravidiel má význam pre SDM systém hlavne v prípade, kedy kapacita firmvéru nepostačuje na pokrytie všetkých pridávaných alebo miera pridávania pravidiel je príliš veľká a radič ich nestíha vkladať do firmvéru. Nevýhodné je tiež vytvárať



Obr. 7.2: Priemerný počet paketov zachytených na jedno pravidlo o toku



Obr. 7.3: Podiel zbytočne zavedených pravidiel o tokoch

pravidlá s príliš malým efektom na spracovanie sieťových dát. Vtedy môže dôjsť k situácii, že réžia SDM radiča spojená s pridávaním pravidla do firmvéru je väčšia ako pravidlom dosiahnuté urýchlenie. Extrémnym prípadom sú pravidlá, ktorých pridanie nemá absolútne žiaden efekt na predspracovanie dát vo firmvéry (nezachytia jediný paket). Zistený podiel takýchto pravidiel v celkovom počte pre simulovanú situáciu je zobrazený na grafe 7.3. Simulácia (modrá) ukazuje až 40 % podiel nepotrebných pravidiel v celkovom počte vytvorených pre nízku hranicu rozhodovania. Ten však rýchlo s rastom hranice výberu klesne na polovicu (už pri výbere 15-tym paketom). Z grafu je tiež vidno, že analytické hodnoty (červená), ktoré zanedbávajú oneskorenie vkladania pravidiel, ukazujú v tomto prípade veľmi skreslenú informáciu.

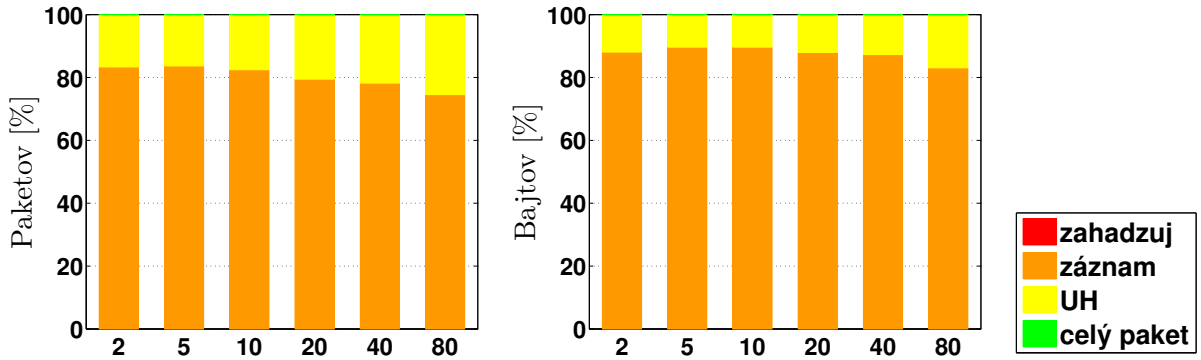
Vyššie popísané grafy ukazujú výhodnosť použitia jednoduchej detekcie ťažkých tokov v SDM na zníženie počtu pridávaných relatívne nepotrebných pravidiel. Zaujímavé bude určite sledovať ako sa tento vplyv prejaví na výkonnosti systému pri nasadení s rôznymi aplikáciami. V nasledujúcich simuláciách preto bude jednoduchá detekcia ťažkých tokov po-

užitá a budú skúmané výsledky pri nastavení niekoľkých rôznych hodnôt hraničnej veľkostí tokov potrebnej na generovanie pravidiel.

7.2 Štandardné monitorovanie tokov

Štandardné monitorovanie tokov bolo bližšie popísané už v teoretickom rozbore práce v sekcii 2.3. Aplikácia vhodná na jeho realizovanie je `flowmonexp` spolu s upraveným vstupným pluginom ako aj v predošlej sekcii. Pri nasadení SDM systému na akceleráciu základného monitorovania tokov postačí ako implicitné predspracovanie pre pakety všetkých tokov zvoliť generovanie UH. Počas behu sú potom aplikáciou pridávané pravidlá pre toky vyžadujúce agregovanie paketov v hardvéri a pravidelný export tokovej UH.

Na základe simulácie nasadenia SDM s aplikáciou na štandardné monitorovanie tokov vznikli grafy na obrázku 7.4. V grafoch je zobrazený podiel použitia jednotlivých spôsobov hardvérového predspracovania na vstupnom sieťovom toku z pohľadu paketov (vľavo) a bajtov (vpravo) pre rôzne použité hraničné veľkosti ťažkého toku. Viditeľné je, že vyše 80 % všetkých paketov tvoriacich dokopy skoro 90 % dátového toku je spracovaných vo firmvéry na záznamy o tokoch. Zvyšok je do softvéru prenášaný vo forme UH.



Obr. 7.4: Podiel použitia spôsobov predspracovania – monitorovanie tokov

V grafoch zobrazené rozdelenia predspracovania sieťových dát vo firmvéri vedú na redukcie objemu toku do softvéru uvedené v tabuľke 7.1. V tabuľke uvedené hodnoty ukazujú schopnosť SDM systému akcelerovať základný monitoring tokov znížením počtu softvérom spracovaných paketov na $\frac{1}{5}$ a objemu dát na $\frac{1}{100}$. Na dosiahnutie týchto výsledkov je potrebné vytvoriť pravidlá pre asi len $\frac{1}{10}$ tokov.

	Dosiahnutá redukcia [%]					
	2	**5**	**10**	**20**	**40**	**80**
Pakety	83,34	83,66	82,45	79,45	78,19	74,55
Bajty	99,14	99,21	99,14	99,02	98,98	98,91
Pravidlá	50,76	71,99	87,84	93,63	96,92	97,76

Tabuľka 7.1: Dosahovaná redukcia sieťových dát – monitorovanie tokov

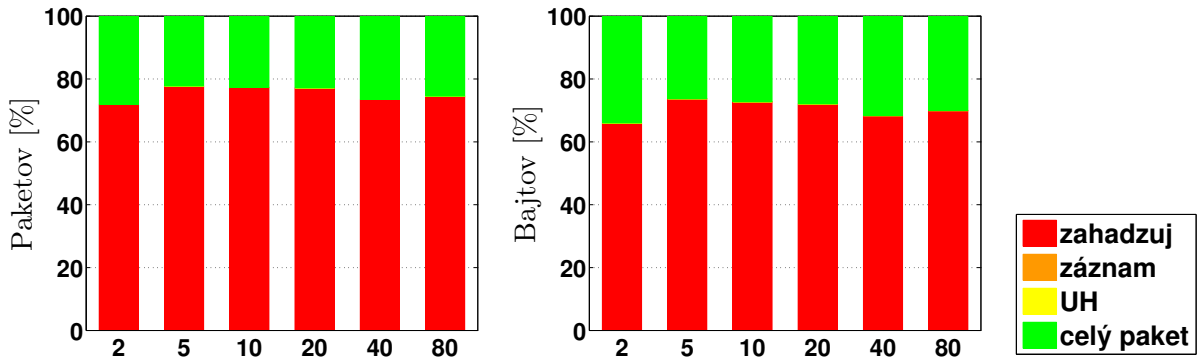
7.3 Analýza aplikačného protokolu HTTP

Protokol HTTP (Hypertext Transfer Protocol) [36] je aplikačným protokolom primárne určeným pre použitie v distribuovaných hypertextových informačných systémoch. Jedná sa o generický, textový, bezstavový protokol fungujúci na princípe požiadaviek a odpovedí (klient – server model). Najrozšírenejšie je v dnešnej dobe jeho použitie na prenos dát v rámci WWW (world wide web). Avšak schopnosť definovať typ a spôsob reprezentácie prenášaných dát umožňuje využiť HTTP aj na rôzne iné účely ako je primárne určený a stále viac aplikácií ho aj využíva. Aj preto sa podľa výsledkov analýzy uvedenej v sekcii 4.1 jedná o najrozšírenejšie používaný aplikačný protokol.

Aplikácia využitá na analýzu HTTP je prevzatá z [37]. Jedná sa o súbor pluginov pre už predstavený `flowmonexp`, ktoré slúžia na extrakciu vybraných zaujímavých informácií z hlavičky HTTP. Extrahované informácie sú potom využité na obohatenie záznamov o sledovaných sieťových tokoch. Implementácia funkčnosti analyzátoru je postavená na konečnom automate vytvorenom použitím programu `flex`. Zaujímavou vlastnosťou fungovania popísaného analyzátoru z pohľadu použitia SMD je, že uchovávané sú vždy len informácie extrahované z prvej HTTP hlavičky v každom toku.

Pri nasadení SDM systému na akceleráciu funkcionality popísaného analyzátoru HTTP je možné už statickými pravidlami zaistiť základné orezanie sieťových dát. Pre samotnú analýzu sú síce potrebné celé HTTP pakety, ale o ostatných nie je potrebné mať žiadnu informáciu. Je preto možné v SDM nastaviť kontext implicitne zahadzujúci všetky pakety mimo HTTP tokov (port 80). Počas behu analyzátoru strácajú preň význam aj pakety HTTP tokov, pre ktoré už extrahoval hlavičku. Preto môže byť pri extrakcii hlavičky priamo pridávané pravidlo do SDM umožňujúce zahadzovanie paketov tohto toku.

Na základe simulácie nasadenia SDM s aplikáciou analyzujúcou HTTP protokol vznikli grafy na obrázku 7.5. V grafoch je zobrazený podiel použitia jednotlivých spôsobov hardvérového predspracovania na vstupnom sieťovom toku z pohľadu paketov (vľavo) a bajtov (vpravo) pre rôzne použité hraničné veľkosti ťažkého toku. Viditeľné je, že skoro 80 % všetkých paketov tvoriacich dokopy vyše 70 % dátového toku je možné už vo firmvéry zahadzovať. Zvyšok však musí byť do softvéru prenášaný v neopracovanej podobe.



Obr. 7.5: Podiel použitia spôsobov predspracovania – analýza HTTP

V grafoch zobrazené rozdelenia predspracovania sieťových dát vo firmvéri vedú na redukcie objemu toku do aplikácie analyzátora uvedené v tabuľke 7.2. V tabuľke uvedené hodnoty ukazujú schopnosť SDM systému akcelerovať samotnú HTTP analýzu znížením

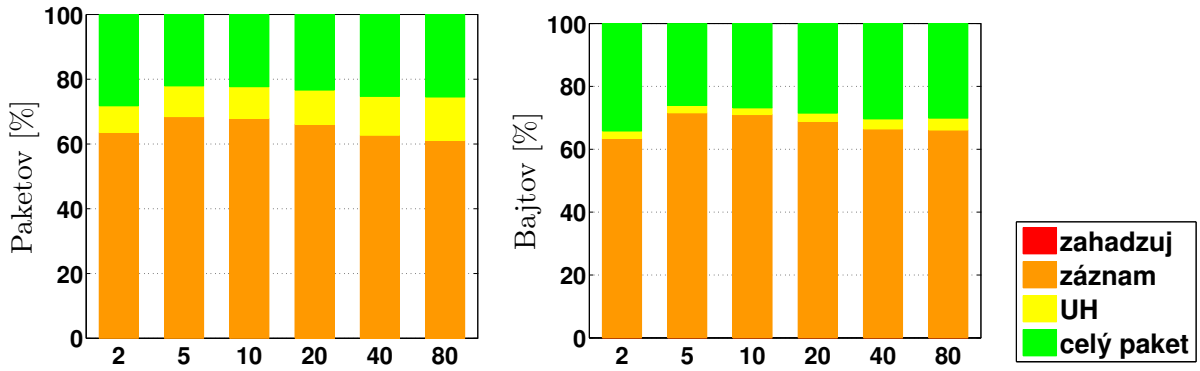
počtu softvérom spracovaných paketov a celkového objemu dát na $\frac{1}{4}$. Na dosiahnutie týchto výsledkov je potrebné vytvoriť pravidlá pre asi len $\frac{1}{20}$ tokov.

	Dosiahnutá redukcia [%]					
	2	**5**	**10**	**20**	**40**	**80**
Pakety	71,77	77,94	77,68	76.65	74.67	74.51
Bajty	65,80	73,91	73,15	71,49	69,66	69,89
Pravidlá	85,07	93,12	96,40	98,15	98,97	99,46

Tabuľka 7.2: Dosahovaná redukcia sieťových dát – analýza HTTP

Okrem skúmania popísaného nasadenia SDM len pri samotnej HTTP analýze je možné simulovať aj prípad realizácie monitorovania tokov obohateného o výsledky HTTP analýzy. V tomto prípade je vhodné pridať k vyššie popísanému kontextu analyzátora aj nezávislý kontext pre štandardné monitorovanie. Ten bude implicitne definovať potrebu každého paketu aspoň v podobe UH. Zároveň bude dynamicky pridávať pravidlá žiadajúce agregovanie paketov vo firmvéri. Vo výsledku je tak umožnené v istej situácii predspracovanie všetkými spôsobmi okrem úplného zahadzovania paketov.

Na základe simulácie nasadenia SDM pri monitorovaní tokov obohatenom o analýzu HTTP vznikli grafy na obrázku 7.6. V grafoch je zobrazený podiel použitia jednotlivých spôsobov hardvérového predspracovania na vstupnom sieťovom toku z pohľadu paketov (vľavo) a bajtov (vpravo) pre rôzne použité hraničné veľkosti ťažkého toku. Viditeľné je, že skoro 70 % všetkých paketov resp. dát je možné spracovávať vo firmvérovej tabuľke tokov. Zo zvyšku musí byť väčšina (celkovo asi 20 %) do softvéru prenášaná v neopracovanej podobe.



Obr. 7.6: Podiel použitia spôsobov predspracovania – analýza HTTP a monitorovanie tokov

	Dosiahnutá redukcia [%]					
	2	**5**	**10**	**20**	**40**	**80**
Pakety	62,97	68,06	67,58	65,89	62,60	61,08
Bajty	65,41	73,47	72,70	71,00	69,11	69,28
Pravidlá	85,27	93,37	96,56	98,24	99,05	99,55

Tabuľka 7.3: Dosahovaná redukcia sieťových dát – analýza HTTP a monitorovanie tokov

V grafoch zobrazené podiely predspracovania sieťových dát vo firmvéri vedú na redukcie objemu toku do aplikácie analyzátora uvedené v tabuľke 7.3. V tabuľke uvedené hodnoty ukazujú schopnosť SDM systému akcelerovať monitorovanie tokov doplnené o HTTP analýzu znížením počtu softvérom spracovaných paketov na necelú $\frac{1}{3}$ a celkového objemu dát na asi $\frac{1}{4}$. Na dosiahnutie týchto výsledkov je potrebné vytvoriť pravidlá pre asi len $\frac{1}{16}$ tokov.

7.4 Analýza aplikačného protokolu DNS

DNS (Domain Name System)[38] predstavuje distribuovaný hierarchický systém mien služieb a zariadení v počítačových sieťach. Najdôležitejšou jeho funkcionalitou je preklad textových doménových mien na numerické IP adresy, podobne ako zlaté stránky. Samotný DNS protokol sa využíva na prístup k popísanému systému, teda na umožnenie zasielanie požiadaviek na preklady adries. DNS využíva k hlavne správy typu požiadavka a odpoveď. Formát jeho správy sa skladá z pevnej hlavičky a variabilného tela.

Aplikácia využitá na analýzu DNS je prevzatá z [39]. Jedná sa o jednoduchú implementáciu rýchleho analyzátora DNS paketov, ktorý ich prevádza do textovej (čitateľnej) podoby. Pôvodná implementácia je určená na použitie nad pcap súbormi. Priloženým skriptom je možné spúšťať ju s programom `tcpdump` na analýzu jeho výstupov.

Pri nasadení SDM systému na akceleráciu funkcionality popísaného analyzátoru DNS je možné už statickými pravidlami zaistiť značnú redukciu objemu sieťových dát. Pre samotnú analýzu sú síce potrebné celé DNS pakety, ale o ostatných nie je potrebné mať žiadnu informáciu. Je preto možné v SDM nastaviť kontext implicitne zahadzujúci všetky pakety mimo DNS tokov (port 53). Z výsledkov analýzy v sekcii 4.1 vyplýva, že DNS toky sú všeobecne veľmi krátke (priemerne len 1,1 paketu na tok). Nie je preto efektívne pridávať dynamické pravidlá do SDM systému.

Na základe simulácie nasadenia SDM pri analýze DNS paketov bola zistená 99,27 % redukcia počtu paketov a 99,84 % redukcia objemu dát a to aj s využitím len statického implicitného pravidla.

Kapitola 8

Záver

Táto diplomová práca je zameraná na návrh a simuláciu systému softvérovo riadenej hardvérovej akcelerácie monitorovacích a bezpečnostných aplikácií pre vysokorýchlostné siete. Hlavný dôraz bol počas návrhu systému kladený na maximalizáciu dosiahnuteľnej miery akcelerácie aplikácií pri jeho nasadení v reálnych vysokorýchlostných sieťach, a tiež na jeho použiteľnosť na rýchlostiach 100 Gb/s.

Riešenie som začal naštudovaním potrebných teoretických znalostí. Medzi ne patrili v prvom rade základné princípy dnešných počítačových sietí a vlastnosti v nich najbežnejšie využívaných protokolov. Na to nadväzoval základný rozbor oblastí monitorovania a bezpečnosti sietí s cieľom. Následne som sa zameral na štúdium základných myšlienok stojacich za novovznikajúcim pohľadom na architektúru sietí označovaným Software Defined Networking. Teoretický rozbor som uzavrel zoznámením sa s exitujúcim riešením hardvérovej akcelerácie pre 1 a 10 Gb/s siete v podobe obecnej platformy Hanic a špecializovanej FlowMon sondy.

Po dokončení rozboru som sa zameral na vytvorenie základného konceptu fungovania navrhovaného akceleračného systému. Pri jeho tvorbe som vychádzal zo spôsobov realizácie naštudovaných existujúcich hardvérových riešení a snažil sa ich vhodne upraviť pre efektívne použitie s monitorovacími a bezpečnostnými aplikáciami v 100 Gb/s sieťach. V dosiahnutí toho mi významne pomohla inšpirácia z myšlienok SDN, navrhnutý systém som preto nazval Software Defined Monitoring. Inšpiráciou bolo striktné rozdelenie funkcionality systému medzi funkčnú hardvérovú časť a inteligentnú softvérovú časť, ktoré spolu úzko spolupracujú. Hardvérová časť systému teda bude schopná realizovať rôzne spôsoby predspracovania a rozdelenia dát pričom ponechá všetku kontrolu tohto predspracovania a pokročilejšie časti úlohy na softvérovú časť, ktorá ju ďalej sprístupní priamo akcelerovaným aplikáciám. Softvérovo riadeným hardvérovým predspracovaním sieťového toku sa docieli aplikačne riadená a hardvérovo akcelerovaná strata informácie podľa aktuálnych potrieb konkrétnej úlohy. Výsledkom je redukcia objemu dát spracovávaných aplikáciou o nezaujímavé informácie vedúca na zvýšenie efektivity a výkonnosti spracovania. Popísaný koncept SDM bol úspešne prezentovaný na konferencii Internet a Technologie 12 v Prahe [40], publikovaný a ocenený na konferencii Student EEICT [41] a publikovaný na konferencii TNC v Maastrichte [42].

Pred pokračovaním k detailnému návrhu funkcionality a realizácie systému SDM som sa ešte pristavil pri skúmaní vhodnosti navrhnutého konceptu pre použitie vo vysokorýchlostných sieťach. Realizoval som to meraním a analýzou vybraných vlastností dátového toku reálnej siete. Skúmané boli hlavne časové a objemové vlastnosti paketov a tokov. Výsledky dosiahnuté analýzou poukazujú na vhodnosť návrhu konceptu systému SDM pre

nasadenie v reálnych sieťach a naznačujú jeho relatívne vysokú efektivitu. Pokračovanie v návrhu SDM spočívalo v podrobnom definovaní funkcionality a architektúry softvérovej a firmvérovej časti systému spolu s ich vzájomným previazaním. Výsledkom je teda podrobný implementačný návrh celého systému. Po dokončení návrhu som prešiel k samotnej tvorbe simulačného modelu celého systému. Základom modelu je plne funkčná implementácia softvérovej časti systému, ktorá je v budúcnosti s minimálnou zmenou použiteľná aj na konfiguráciu samotného reálneho firmvéru. Momentálne je ale nastavená na konfigurovanie vytvoreného jednoduchého simulačného modelu firmvérovej časti.

Použitím simulačného modelu bolo sledované správanie sa navrhnutého SDM systému pri jeho nasadení v rôznych situáciách. Najprv bola simulovaná všeobecná schopnosť systému meniť spôsob predspracovania tokov na základe pridávania dynamických pravidiel pre toky za behu. Z výsledkov simulácií vyplynulo, že systém je schopný pridávanými pravidlami ovplyvniť až 80 % z celkového počtu paketov pri potrebe zavedenia pravidiel len pre 5 % všetkých tokov, čo vedie na priemerne 500 paketov zachytených každým pridaným pravidlom. Ďalšie realizované simulácie fungovania SDM systému boli už zamerané na dosiahnuteľnú redukciu sieťového toku dát pri jeho nasadení s rôznymi aplikáciami. Prvou testovanou situáciou bolo použitie na podporu základného monitorovania tokov. Dosiahnuteľná redukcia počtu paketov sa pohybovala okolo 80 %, pričom redukcia objemu dát bola až 99 %. A to všetko pri potrebe vytvoriť pravidlo len pre 7 % tokov. Pri simulovaní nasadenia SDM s analyzátorom HTTP protokolu bola dosiahnutá trochu nižšia, avšak stále relatívne vysoká miera redukcie – asi 75 % počtu paketov a 73 % objemu dát resp. 67 % a 72 % pri súbežnom behu s monitorovaním tokov. Nakoniec bolo simulované použitie systému s DNS analyzátorom kedy bola dosiahnutá vyše 99 % redukcia sieťového toku.

Nad rámec zadania som navyše začal s implementáciou firmvérovej časti systému. Ako prvý som vytvoril modulárny vysoko konfigurovateľný analyzátor hlavičiek rámcov v jazyku VHDL, ktorý je schopný dosahovať priepustnosť v rozsahu desiatok až stoviek Gb/s a je jednoducho nastaviteľný pre použitie v rôznych aplikáciách a podporu rôznych protokolov. Hlavným využitým konceptom je plne nastaviteľné zreťazenie procesnej linky, ktoré umožňuje nájdenie najlepšieho riešenia podľa konkrétnych požiadaviek. Analyzátor zaberá len 1,19 % zdrojov na FPGA čipe Virtex-7 870HT pri dosiahnutí priepustnosti 100 Gb/s potrebnej na použitie v SDM firmvéri. Zároveň je možné ho nastaviť až na priepustnosť vyše 400 Gb/s pri využití len 4,88 % zdrojov spomenutého čipu. Popísaný návrh modulárneho hardvérového analyzátora hlavičiek paketov bol publikovaný na odbornej ACM/IEEE konferencii ANCS v Austine [33].

Ďalej som nad rámec zadania navrhol a implementoval prototyp hlavnej časti klasifikačnej jednotky potrebnej pre SDM. Vytvorený prototyp je postavený na využití haš tabuľky s plne hardvérovo implementovaným princípom kukučieho hašovania. Priemerné dosahované zaplnenie kapacity vytvorenej haš tabuľky sa pohybuje okolo 90 %. Priepustnosť je zároveň dostatočná na zvládnutie spracovania aj najkratších rámcov na plnej rýchlosti 100 Gb/s. Na plnohodnotné použitie v klasifikátore SDM firmvéru je však ešte potrebné zvýšiť kapacitu vytvorenej haš tabuľky implementovaním podpory uloženia jej častí aj v externej pamäti a nielen v interných pamätiach na FPGA čipe. Vytvorená haš tabuľka s kukučím hašovaním je však už aj v terajšej podobe využitá v rámci hardvérovo akcelerovaných 1 a 10 Gb/s sond pre systém legálneho odpočúvania vytváraný na FIT VUT v Brne v rámci projektu Sec6net [35].

Plánovaným budúcim pokračovaním práce je hlavne dokončenie firmvéru navrhnutého SDM systému. Potrebné je napríklad už spomínané doplnenie podpory práce s externou pamäťou do prototypu klasifikačnej jednotky založenej na kukučom hašovaní. Takisto je

nutné kompletne navrhnúť a implementovať ďalšie jednotky firmvéru: správcu záznamov, exportér, pamäťový arbiter a softvérový prístup. Navrhnutú architektúru je tiež možné rozširovať na podporu akcelerácie ďalšej špecifickej funkcionality. Jednou z možností je napríklad predradenie jednoduchého filtru IP adries pred klasifikačnú jednotku na podporu základnej LI funkcionality. Ďalšou možnosťou je rozšírenie funkcionality správcu tokov na podporu počítania rôznych pokročilejších štatistík okrem základného NetFlow.

Literatúra

[1] Liberouter / Cesnet TMC group. *Hanic* [online]. Updated: 13 July 2012 [cit. 8. január 2013]. Dostupné na URL: <https://www.liberouter.org/?page_id=821>.

[2] Cisco Networking Academy. *CCNA Exploration Course Booklet: Network Fundamentals, Version 4.0*. Indianapolis, USA: Cisco Press, September 2009. Course Booklets. ISBN 978-1-58713-243-8.

[3] Kekely, L. *Hardwarová akcelerace operace hledání nejdelšího společného prefixu*. Brno: FIT VUT v Brně, 2011. 64 s. Bakalářská práce. Dostupné na URL: <http://www.fit.vutbr.cz/study/DP/all.php?id=12737>.

[4] Matoušek, P. *Sít'ové aplikace a správa sítí: Architektura sítí, adresování, testování*. Brno: FIT VUT v Brně, 2010.

[5] Postel, J. *Internet Protocol* [RFC 791]. September 1981. Dostupné na URL: <http://www.ietf.org/rfc/rfc791.txt>.

[6] Deering, S. a Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification* [RFC 2460]. December 1998. Dostupné na URL: <http://www.ietf.org/rfc/rfc2460.txt>.

[7] Hinden, R. a Deering, S. *IP Version 6 Addressing Architecture* [RFC 4291]. February 2006. Dostupné na URL: <http://www.ietf.org/rfc/rfc4291.txt>.

[8] Postel, J. *Transmission Control Protocol* [RFC 793]. September 1981. Dostupné na URL: <http://www.ietf.org/rfc/rfc793.txt>.

[9] *User Datagram Protocol* [RFC 768]. August 1980. Dostupné na URL: <http://www.ietf.org/rfc/rfc768.txt>.

[10] Cecil, A. *A Summary of Network Traffic Monitoring and Analysis Techniques*. Supervisor: Prof. Raj Jain. St. Louis, USA: CSE WU in St. Louis, 2006. Dostupné na URL: <http://www1.cse.wustl.edu/~jain/cse567-06/net_monitoring.htm>.

[11] Uithol, M. a Kooten, V. van. *Network monitoring based on flow measurement techniques*. The Netherlands: University of Twente, 2005.

[12] Varga, L. *Flexibilní měření toků na síti*. Brno: FIT VUT v Brně, 2010. S. 4–19. Diplomová práce. Dostupné na URL: <http://www.fit.vutbr.cz/study/DP/all.php?id=9407>.

[13] Matoušek, P. *Síťové aplikace a správa sítí: Měření provozu na síti pomocí NetFlow*. Brno: FIT VUT v Brně, 2012.

[14] Claise, B. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954 (Informational)]. October 2004. Dostupné na URL: <http://www.ietf.org/rfc/rfc3954.txt>.

[15] Claise, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information* [RFC 5101 (Proposed Standard)]. January 2008. Dostupné na URL: <http://www.ietf.org/rfc/rfc5101.txt>.

[16] Northcutt, S., Zeltser, L., Winters, S. et al. *Inside Network Perimeter Security*. Second Edition. Indianapolis, USA: Sams Publishing, March 2005. 768 s. ISBN 0-672-32737-6.

[17] Kurose, J. F. a Ross, K. W. *Computer Networking: A Top-Down Approach Featuring the Internet*. First Edition. Boston, USA: Addison Wesley, July 2000. 712 s. ISBN 0-201-47711-4.

[18] Cisco DocWiki. *Internetworking Technology Handbook: Security Technologies* [online]. Updated: 16 October 2012 [cit. 8. január 2013]. Dostupné na URL: <http://docwiki.cisco.com/wiki/Security_Technologies>.

[19] Kořenek, J. *Rychlé vyhledávání regulárního výrazů s využitím technologie FPGA*. Brno: FIT VUT v Brně, 2010. 100 s. Disertační práce. Dostupné na URL: <http://www.fit.vutbr.cz/study/DP/PD.php?id=162>.

[20] Nechvatal, J. *Public-Key Cryptography*. Gaithersburg, Maryland, USA: NIST, 1991. 158 s.

[21] ONF Market Education Committee. *Software-Defined Networking: The New Norm for Networks*. Palo Alto, California, USA: Open Networking Foundation, 2012. ONF White Paper.

[22] The Liberouter Project Team. *Hashing Network Interface Card Handbook* [online]. Version 2.6.1. Updated: 4 October 2011 [cit. 8. január 2013]. Dostupné na URL: <http://www.liberouter.org/package_releases/hanic-current/hanic-combov2-handbook.html>.

[23] Čeleda, P., Kováčik, M., Koníř, T. et al. *FlowMon Probe*. Praha: CESNET, 2006. 15 s. CESNET technical report 31/2006.

[24] IANA. *Protocol Numbers* [online]. Updated: 17 February 2013 [cit. 8. máj 2013]. Dostupné na URL: <http://www.iana.org/assignments/protocol-numbers/>.

[25] IANA. *Service Name and Transport Protocol Port Number Registry* [online]. Updated: 2 May 2013 [cit. 8. máj 2013]. Dostupné na URL: <http://www.iana.org/assignments/service-names-port-numbers/>.

[26] Žádník, M. *Optimalizace sledování síťových toků*. Brno: FIT VUT v Brně, 2013. 119 s. Disertační práce.

[27] Hall, B. *Beej's Guide to Unix IPC* [online]. Updated: 15 December 2010 [cit. 20. apríl 2013]. Dostupné na URL: <http://beej.us/guide/bgipc/output/html/multipage/>.

[28] The Open Group. *The Open Group Base Specifications: write*. 2004. Dostupné na URL: <http://pubs.opengroup.org/onlinepubs/009695399/functions/write.html>.

[29] Microsoft Developer Network. *Lockless Programming Considerations for Xbox 360 and Microsoft Windows* [online]. Updated: 22 November 2012 [cit. 20. apríl 2013]. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/ee418650%28VS.85%29.aspx>.

[30] Free Software Foundation. *Using the GNU Compiler Collection: Atomic Builtins* [online]. Updated: 14 February 2007 [cit. 20. apríl 2013]. Dostupné na URL: <http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/Atomic-Builtins.html>.

[31] ISO. *ISO/IEC 9899:2011 Information technology — Programming languages — C*. Geneva, Switzerland: International Organization for Standardization, 2011. S. 17–21.

[32] Attig, M. a Brebner, G. J. 400 Gb/s Programmable Packet Parsing on a Single FPGA. In *Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. Brooklyn, NY, USA: IEEE, 2011. S. 12–23.

[33] Kekely, L., Kořenek, J. a Puš, V. Low-latency modular packet header parser for FPGA. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*. New York, NY, USA: ACM, 2012. S. 77–78. Dostupné na URL: <http://doi.acm.org/10.1145/2396556.2396571>. ISBN 978-1-4503-1685-9.

[34] Pagh, R. a Rodler, F. F. Cuckoo Hashing. In Meyer auf der Heide, F. (ed.). *Algorithms – ESA 2001*. Aarhus, Denmark: Springer, 2001. S. 121–133. Lecture Notes in Computer Science, sv. 2161. ISBN 3-540-42493-8.

[35] Kekely, L. a Žádník, M. *Hardwarově akcelerovaná sonda pro legální odposlechy*. 2012. 18 s. Dostupné na URL: <http://www.fit.vutbr.cz/research/view_pub.php?id=10214>.

[36] Fielding, R., Gettys, J., Mogul, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [RFC 2616 (Draft Standard)]. 1999. Dostupné na URL: <http://www.ietf.org/rfc/rfc2616.txt>.

[37] Šíma, T., Velan, P. a Čeleda, P. *FlowMon – Plugins for HTTP Monitoring* [online]. Updated: 20 December 2012 [cit. 31. apríl 2013]. Dostupné na URL: <http://dior.ics.muni.cz/ velan/flowmon-input-http/>.

[38] Mockapetris, P. *Domain names - concepts and facilities* [RFC 1034 (Standard)]. 1987. Dostupné na URL: <http://www.ietf.org/rfc/rfc1034.txt>.

[39] Ferrell, P. *A Fast Parser for DNS Pcap Data* [online]. Updated: 20 March 2013 [cit. 20. máj 2013]. Dostupné na URL: <https://github.com/pflarr/dns_parse>.

[40] Friedl, S., Kekely, L. a Puš, V. *Monitorovanie sietí na rýchlosti 100 Gb/s: Internet a Technologie 12*. 2012. Dostupné na URL: <http://www.nic.cz/public_media/IT12/prezentace/IT12_Lukas_Kekely.pdf>.

[41] Kekely, L. Hardware Acceleration of Netwrok Security and Monitoring Applications. In *Proceedings of the 19th Conference STUDENT EEICT 2013*. [b.m.]: Brno University of Technology, 2013.

[42] Kekely, L. a Puš, V. Software Defined Monitoring: A New Approach to Network Traffic Monitoring. In *TNC*. Maastricht, Netherlands: TERENA, 2013.

Dodatok A

Dodatočné grafy k analýze sieťového toku

A.1 Časové vlastnosti tokov

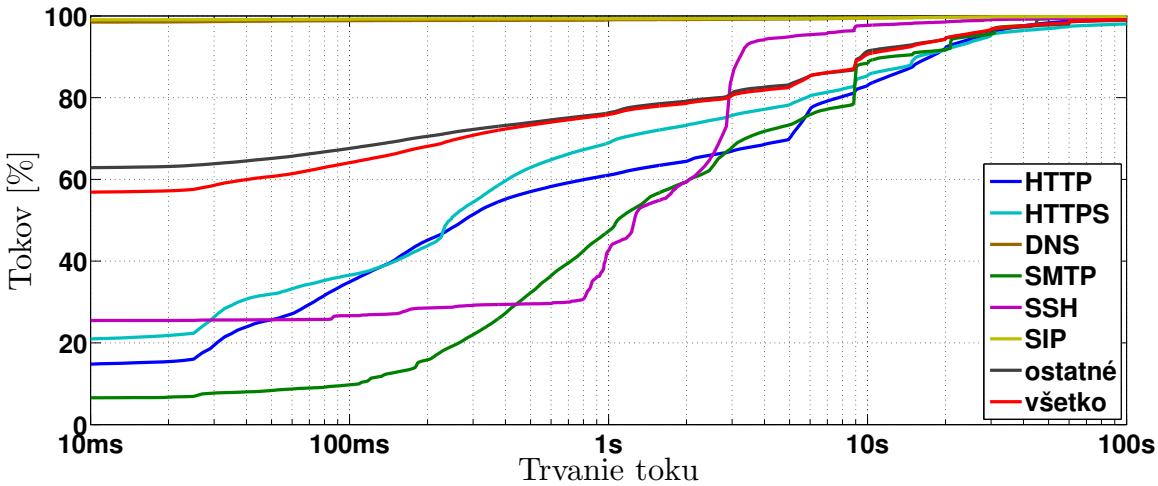
Grafy na obrázkoch A.1 a A.2 predstavujú smerovo závislé varianty grafu 4.4. Podobne grafy na obrázkoch A.3 a A.4 predstavujú smerovo závislé varianty grafu 4.6.

A.2 Objemové vlastnosti tokov

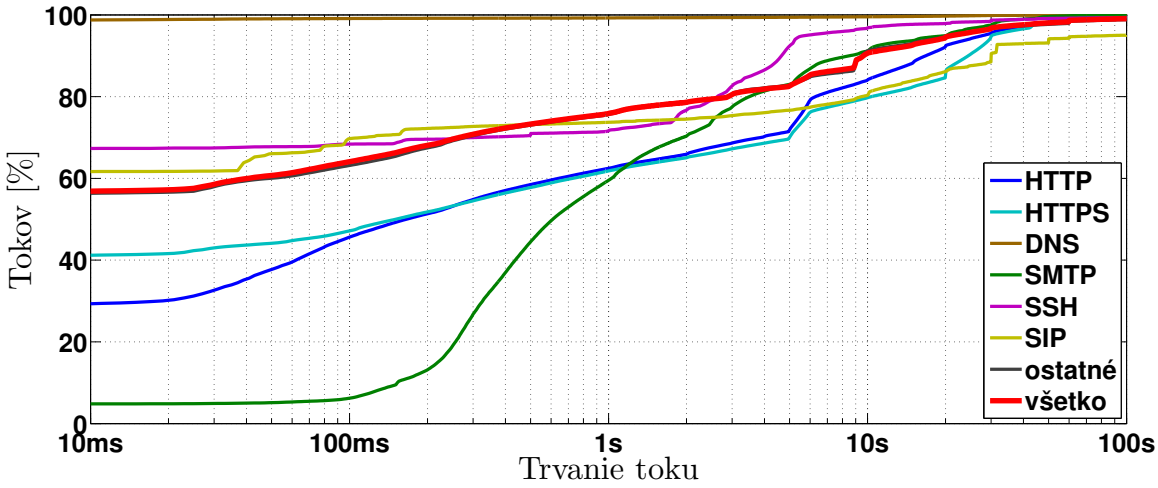
Grafy na obrázkoch A.5 a A.6 predstavujú smerovo závislé varianty grafu 4.8. Podobne grafy na obrázkoch A.7 a A.8 predstavujú smerovo závislé varianty grafu 4.9.

A.3 Detekovanie ťažkých tokov

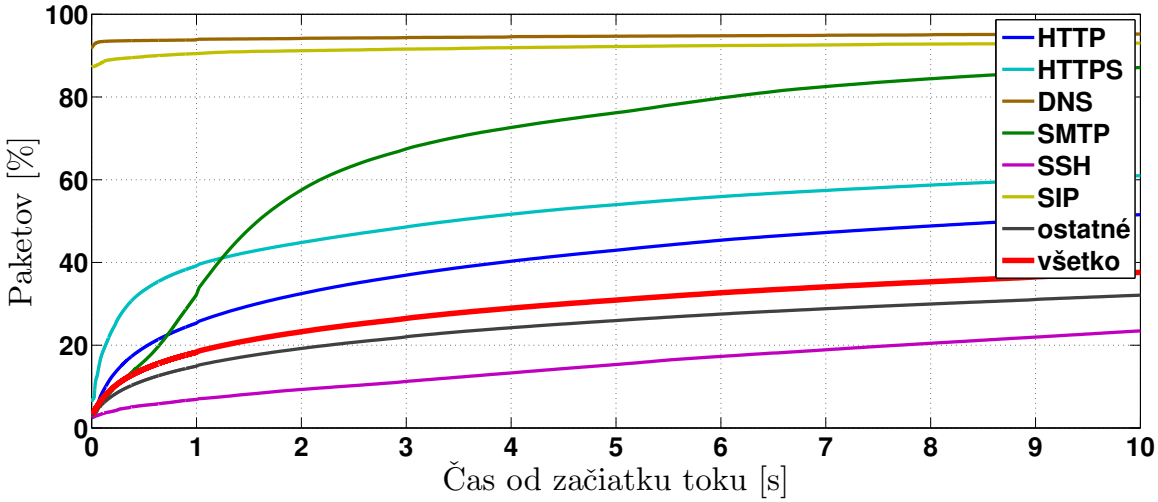
Grafy na obrázkoch A.9 a A.10 predstavujú smerovo závislé varianty grafu 4.12. Podobne grafy na obrázkoch A.11 a A.12 predstavujú smerovo závislé varianty grafu 4.14.



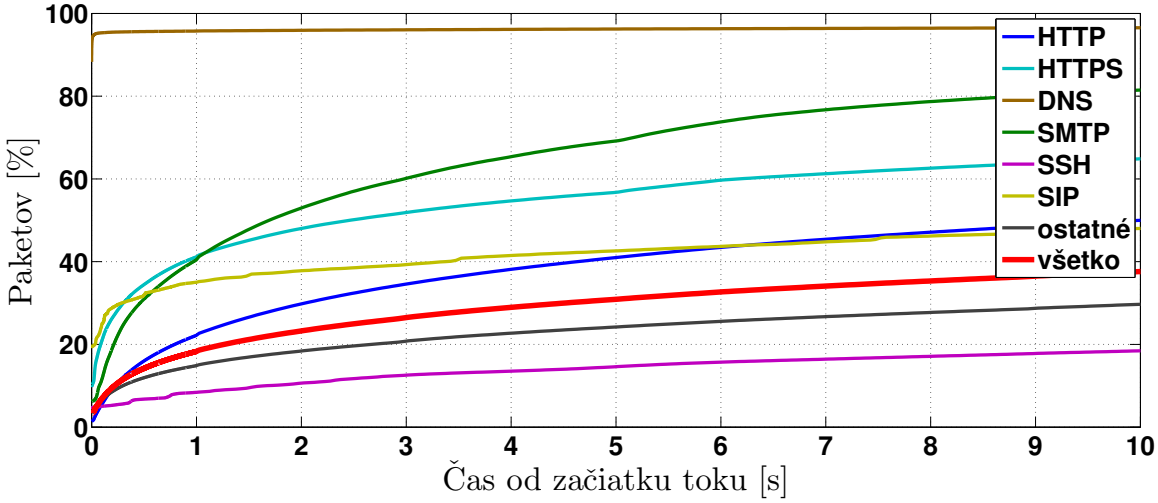
Obr. A.1: Distribučné funkcie trvania tokov – cieľová služba



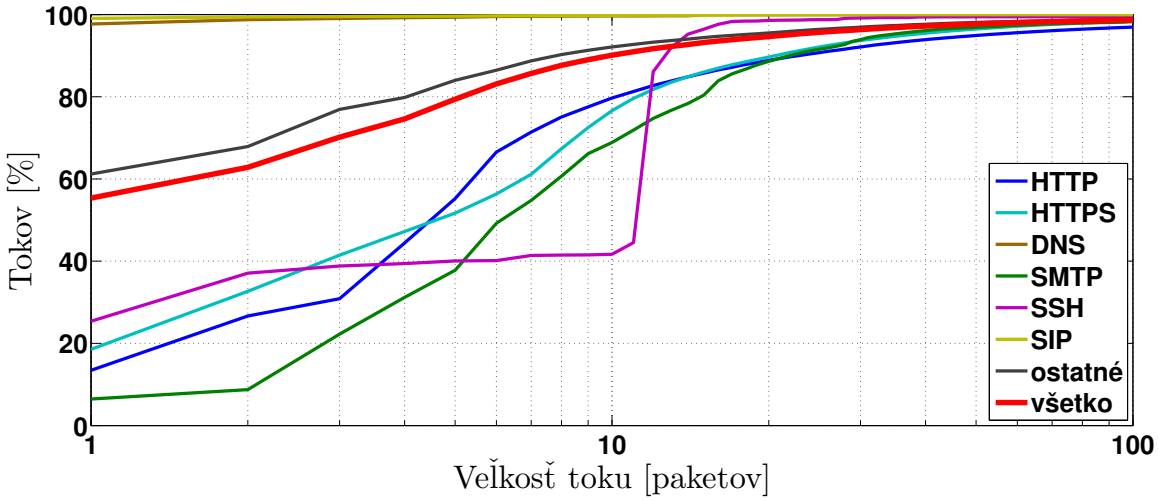
Obr. A.2: Distribučné funkcie trvania tokov – zdrojová služba



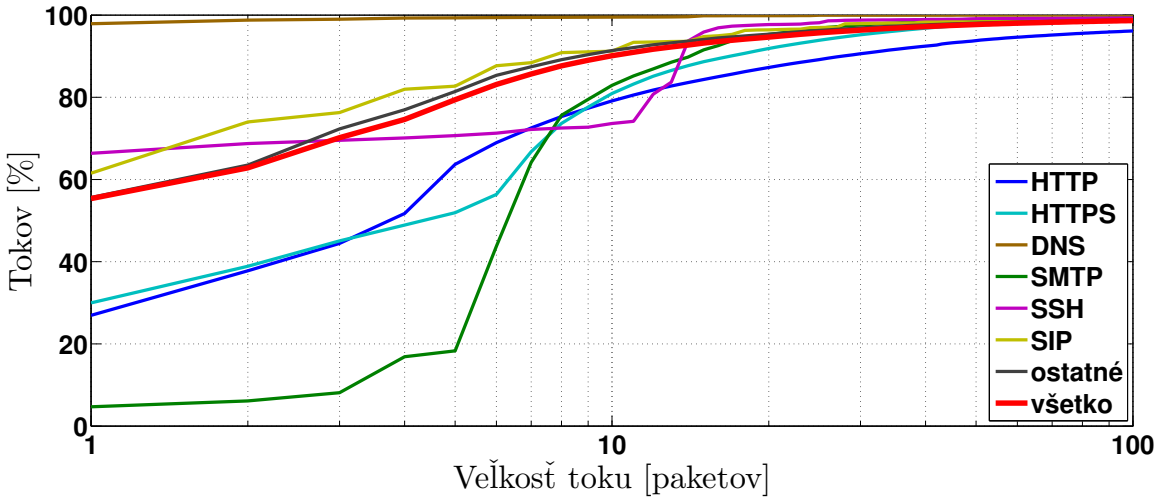
Obr. A.3: Distribučné funkcie časov výskytu paketov v tokoch – cieľová služba



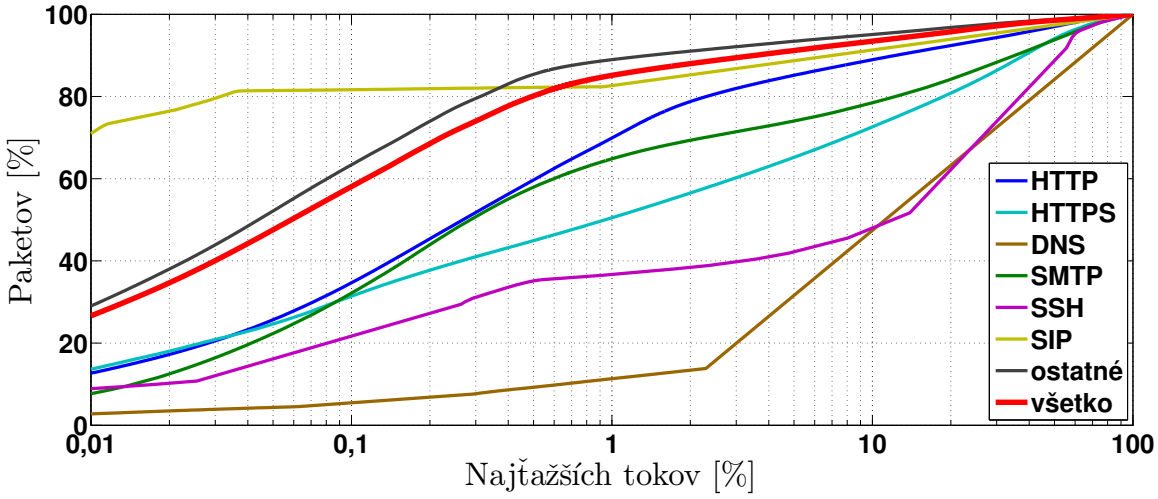
Obr. A.4: Distribučné funkcie časov výskytu paketov v tokoch – zdrojová služba



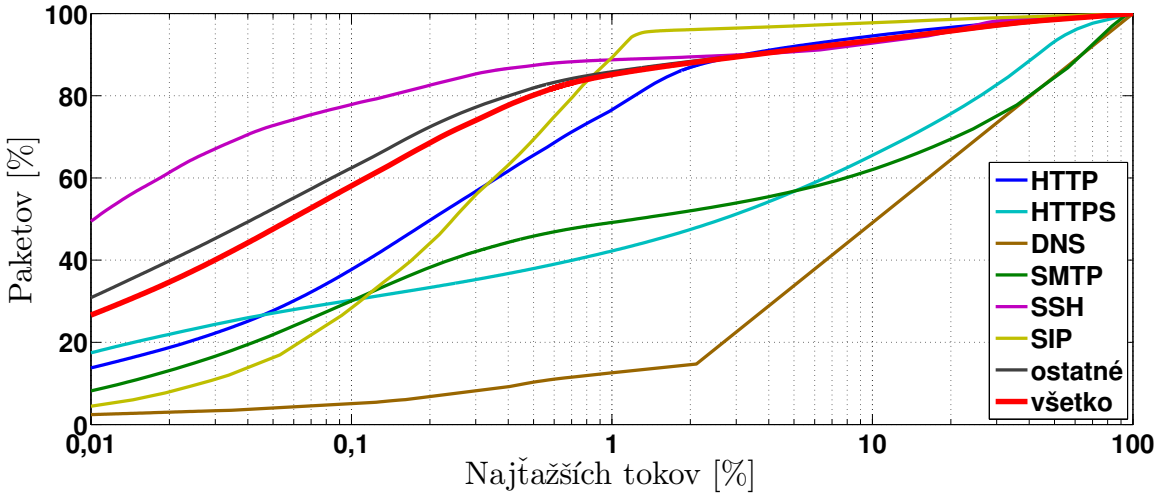
Obr. A.5: Distribučné funkcie veľkosti tokov – cieľová služba



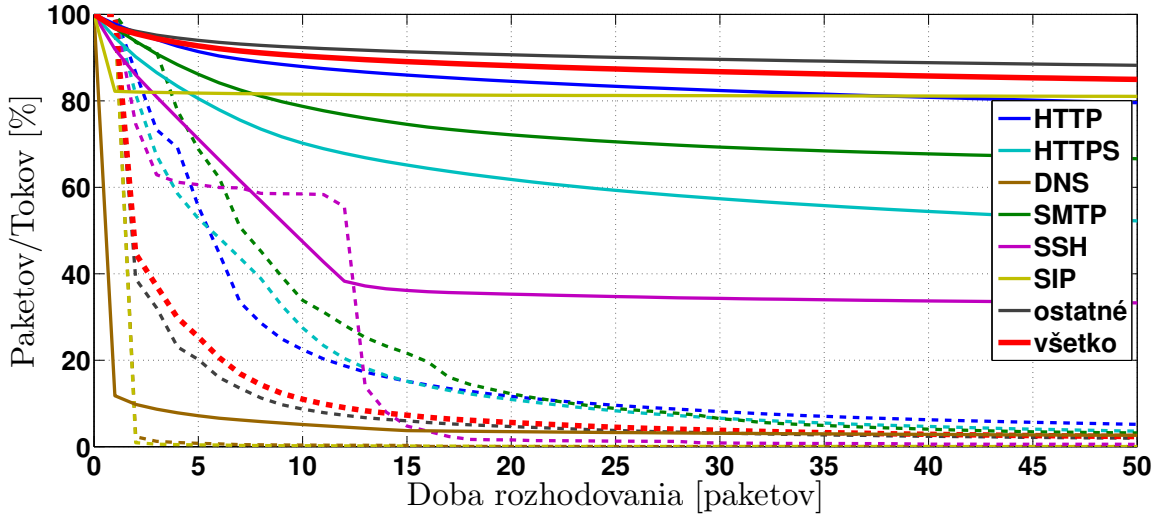
Obr. A.6: Distribučné funkcie veľkosti tokov – zdrojová služba



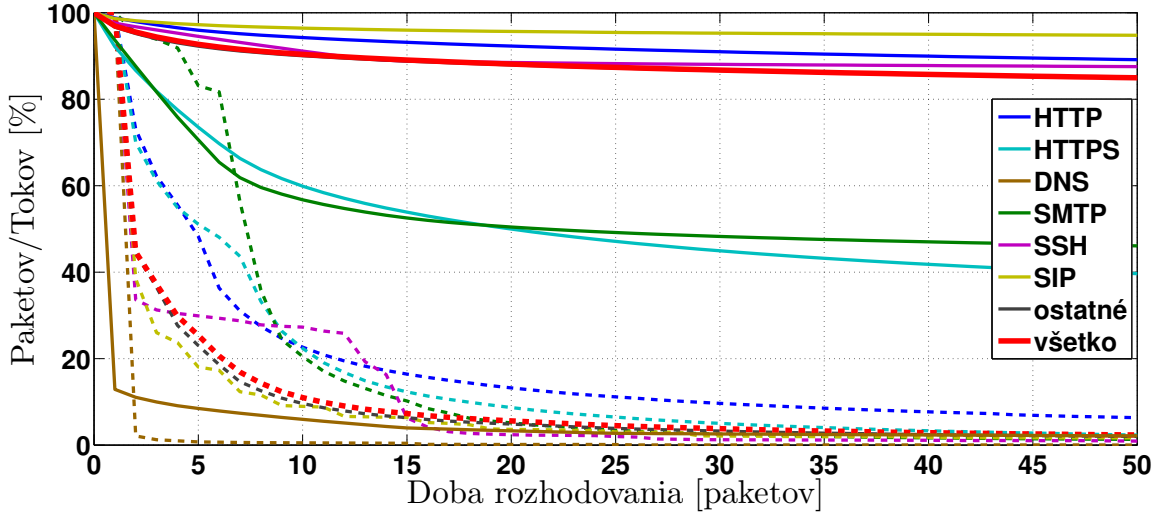
Obr. A.7: Heavy-tailed rozloženie počtu paketov v tokoch – cieľová služba



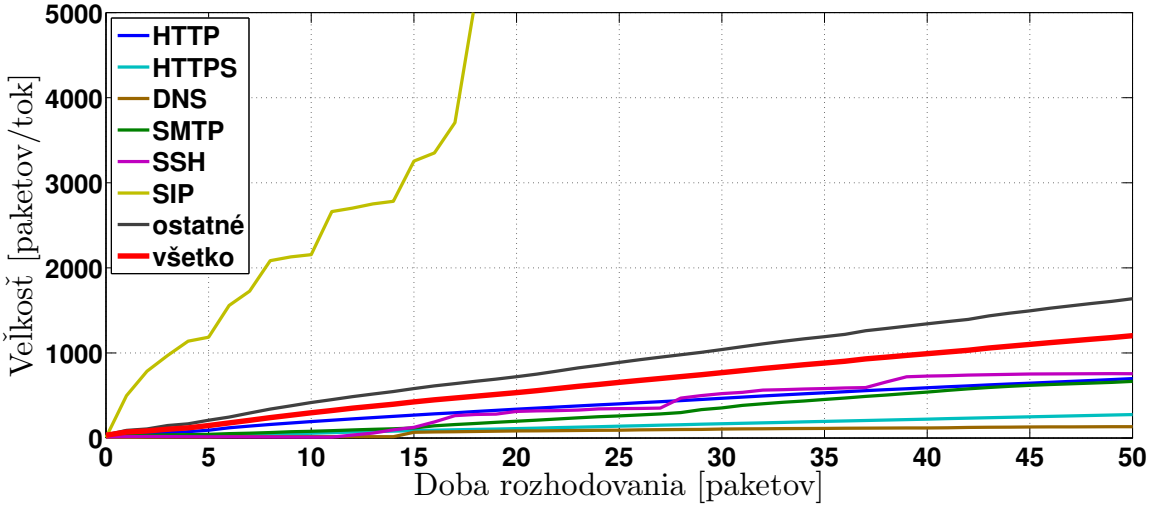
Obr. A.8: Heavy-tailed rozloženie počtu paketov v tokoch – zdrojová služba



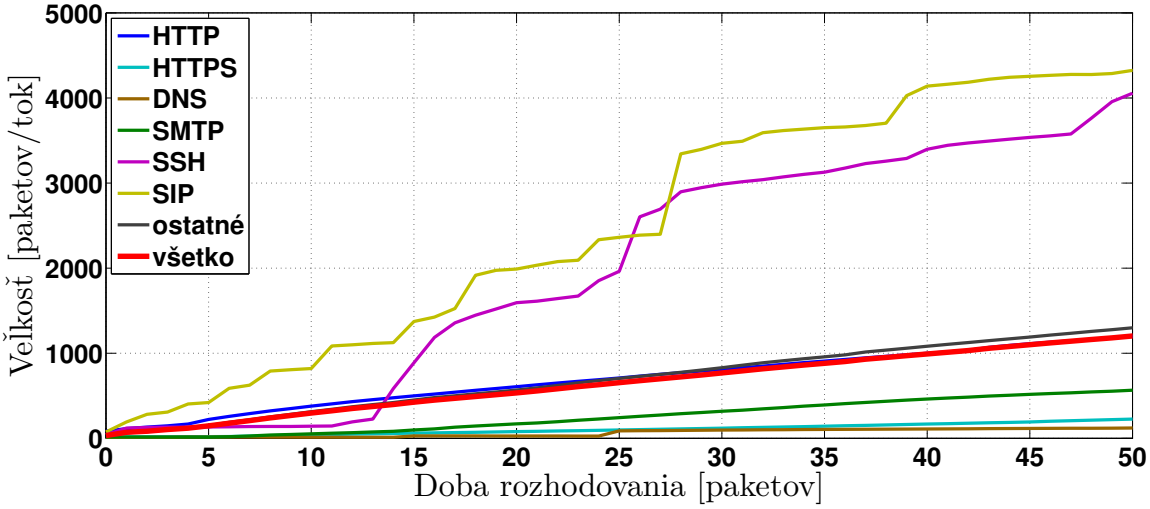
Obr. A.9: Výsledky detekcie ťažkých tokov naivnou metódou – cieľová služba



Obr. A.10: Výsledky detekcie ťažkých tokov naivnou metódou – zdrojová služba



Obr. A.11: Priemerný počet paketov na tok pri naivnej metóde – cieľová služba

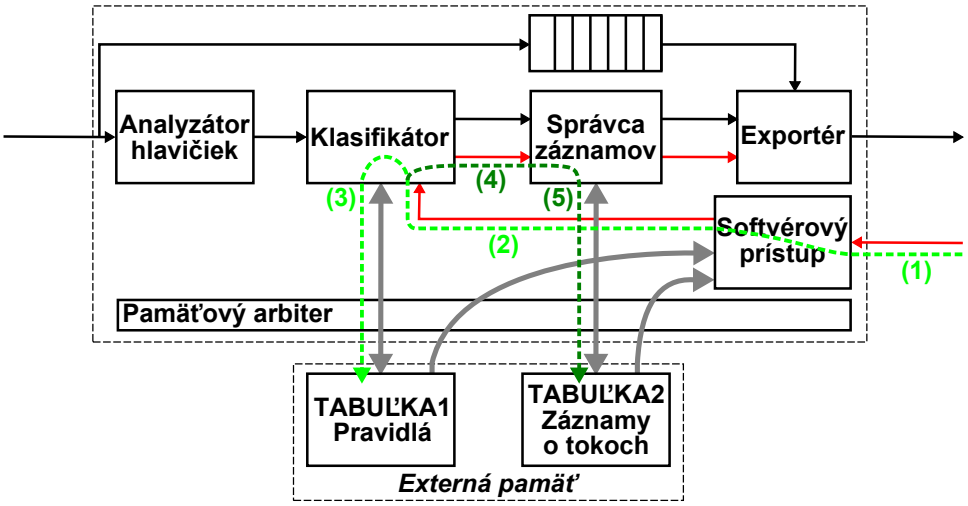


Obr. A.12: Priemerný počet paketov na tok pri naivnej metóde – zdrojová služba

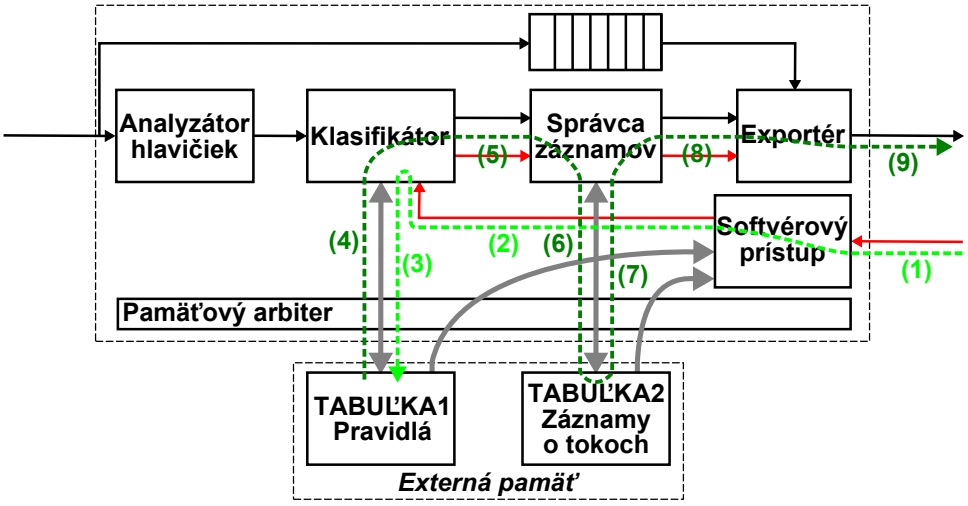
Dodatok B

Ilustrácie operácií firmvéru SDM

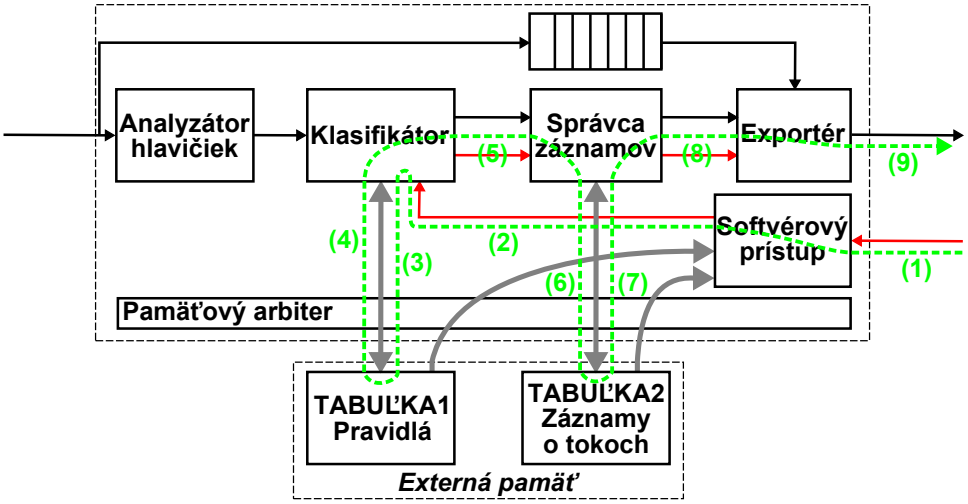
Obrázky B.1 až B.5 ilustrujú fungovanie operácií popísaných na konci sekcie 5.1.



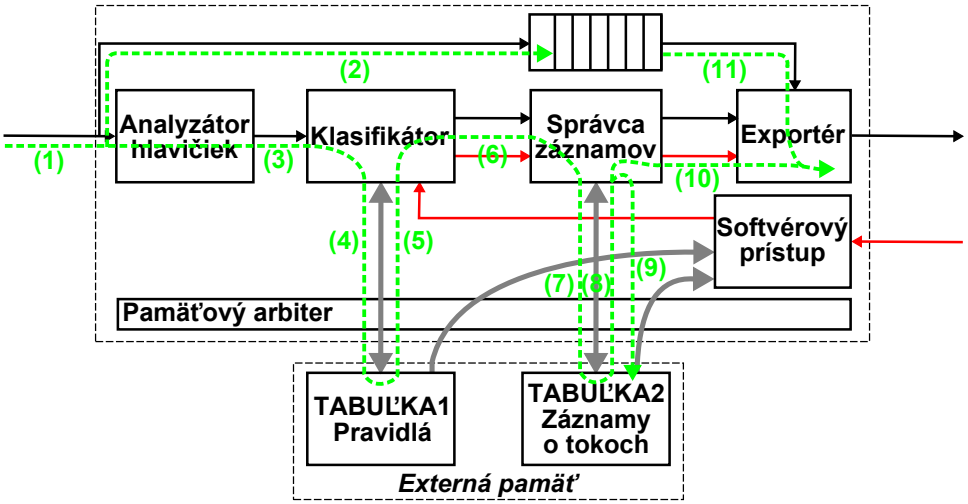
Obr. B.1: Ilustrácia postupu pridania pravidla



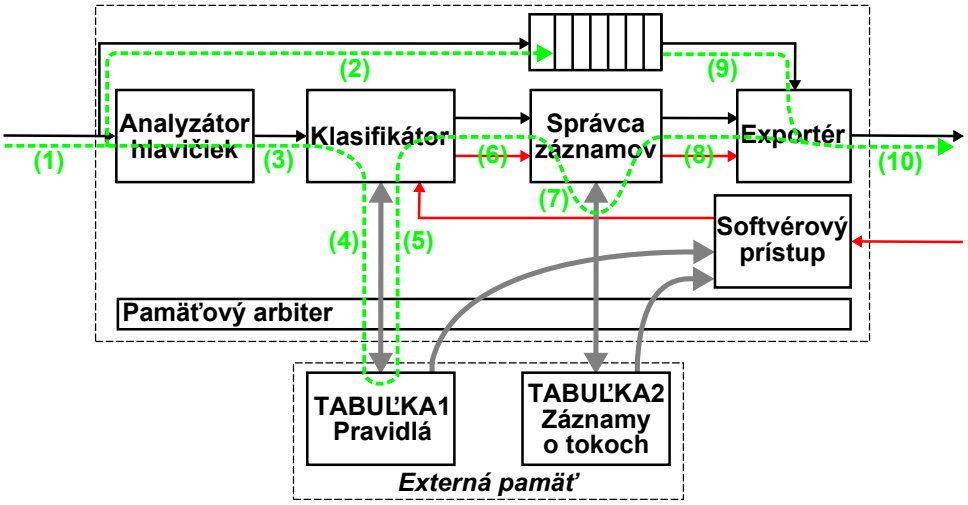
Obr. B.2: Ilustrácia postupu odobrania pravidla



Obr. B.3: Ilustrácia postupu exportu a nulovania záznamu o toku



Obr. B.4: Ilustrácia spracovania rámca vedúceho na úpravu záznamu o toku

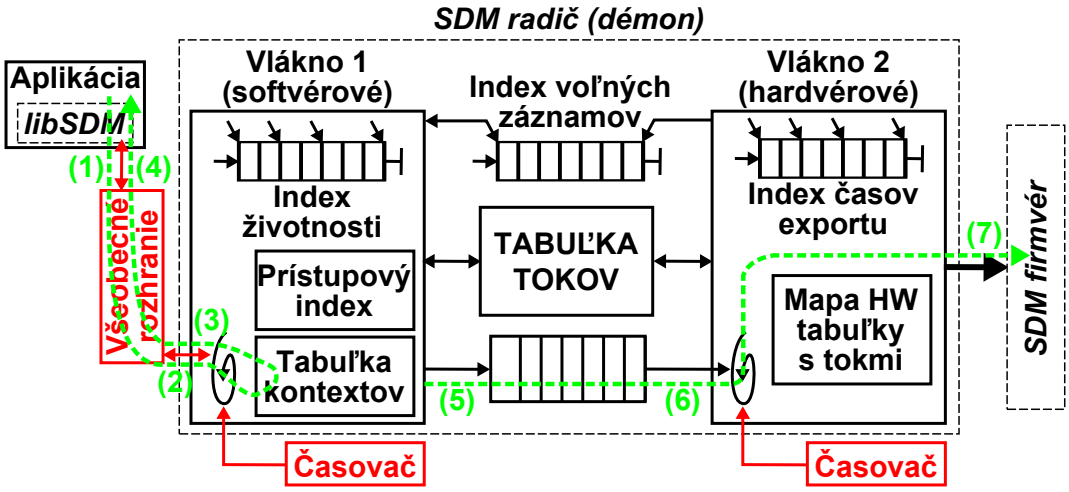


Obr. B.5: Ilustrácia spracovania rámca vedúceho na poslanie dát do SW

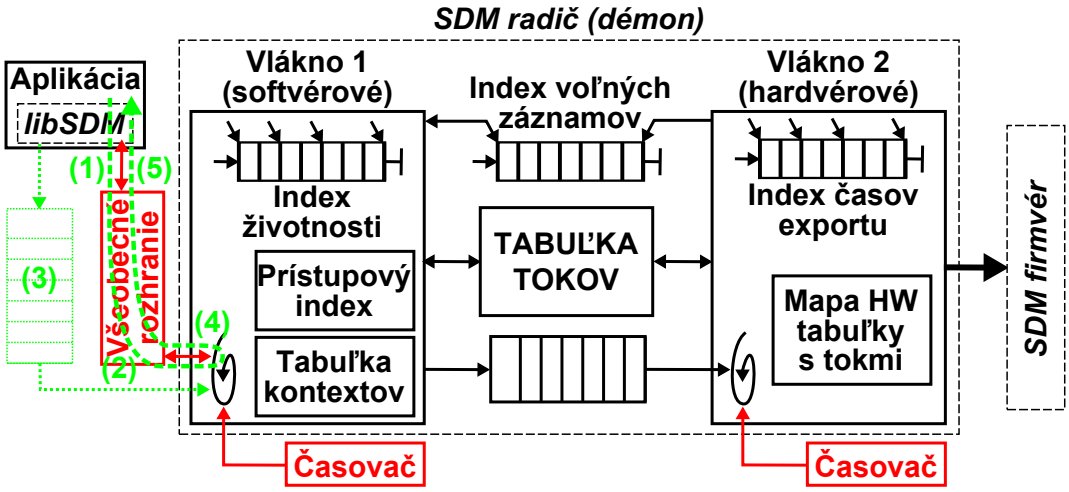
Dodatok C

Ilustrácie operácií radiča SDM

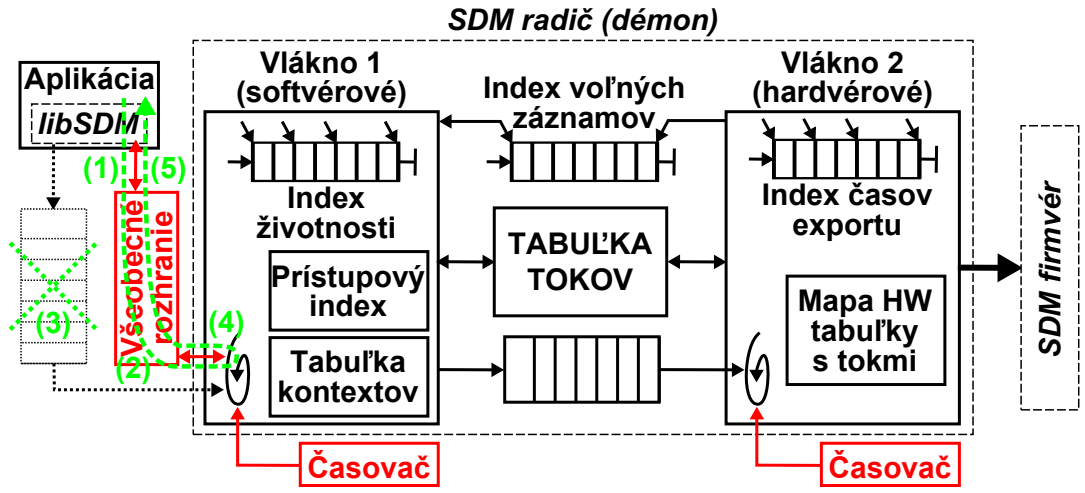
Obrázky C.1 až C.8 ilustrujú fungovanie operácií popísaných na konci sekcie 5.2.



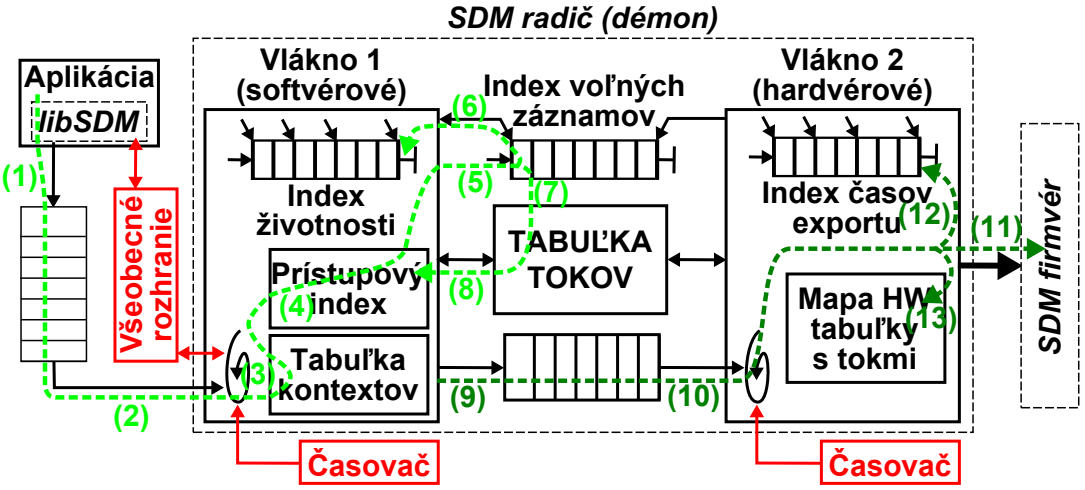
Obr. C.1: Ilustrácia postupu radiča SDM pri zmene v kontextoch



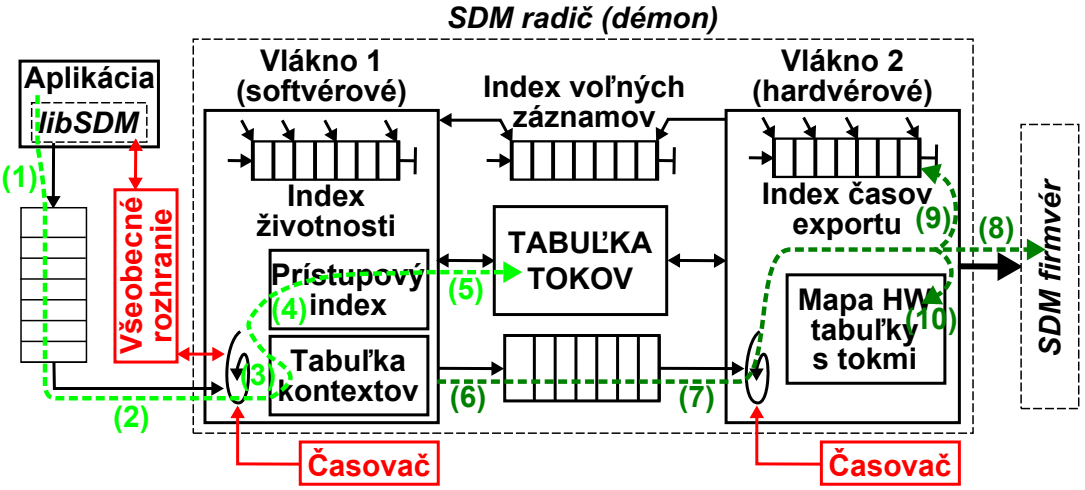
Obr. C.2: Ilustrácia postupu radiča SDM pri pridaní zdroja pravidiel



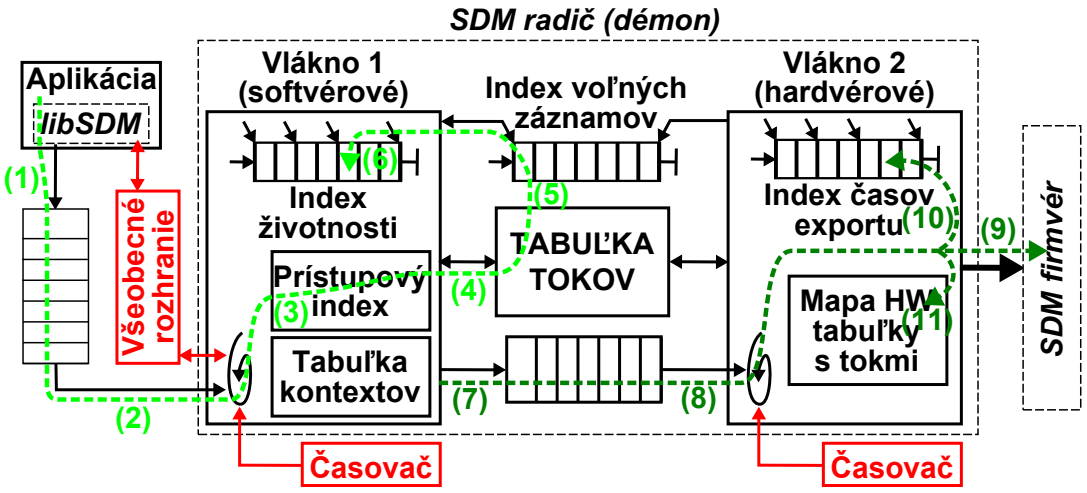
Obr. C.3: Ilustrácia postupu radiča SDM pri rušení zdroja pravidiel



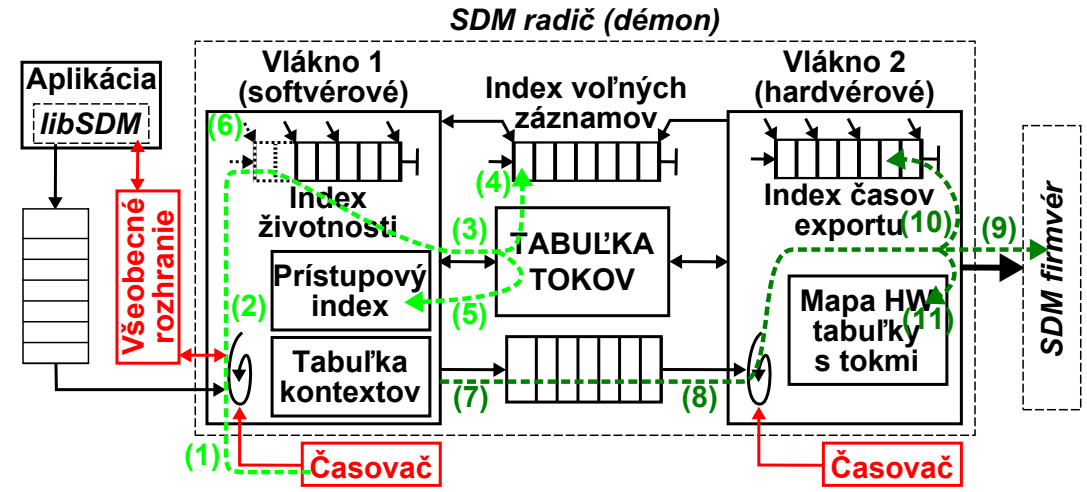
Obr. C.4: Ilustrácia postupu radiča SDM pri pridávaní nového pravidla



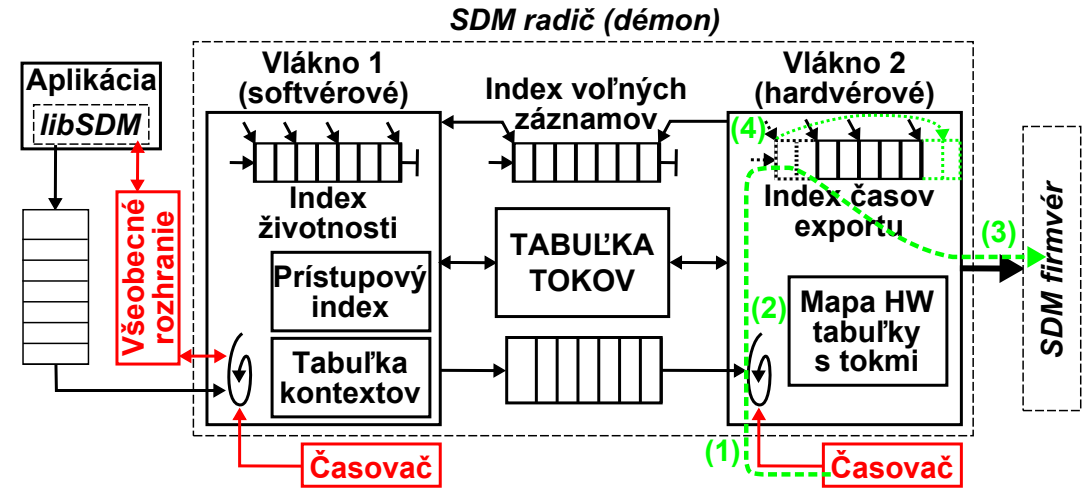
Obr. C.5: Ilustrácia postupu radiča SDM pri úprave pravidla



Obr. C.6: Ilustrácia postupu radiča SDM pri rušení pravidla



Obr. C.7: Ilustrácia postupu radiča SDM pri vypršaní platnosti záznamov



Obr. C.8: Ilustrácia postupu radiča SDM pri pravidelnom exporte záznamov