Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Redukcia informácií o IP tokoch za účelom zníženia záťaže monitorovacích systémov

Diplomová práca

2014 **Bc. Samuel Tremko**

Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Redukcia informácií o IP tokoch za účelom zníženia záťaže monitorovacích systémov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Peter Feciľak, PhD.
Konzultant:	Ing. Adrián Pekár

Košice 2014 **Bc. Samuel Tremko**

Abstrakt v SJ

Táto práca je venovaná technikám na redukciu informácií o IP tokoch za účelom zníženia záťaže monitorovacieho systému SLAmeter. Implementácia týchto techník je realizovaná v nástroji MyBeem. V úvode práce je vykonaná analýza nástroja SLAmeter, ktorý pre účely merania rôznych prevádzkových charakteristík používa nástroj MyBeem. Analýze sú podrobené aj techniky na redukciu informácií o IP tokoch a to najmä rôzne techniky agregácie. V tejto práci sú navrhnuté, implementované a overené úpravy nástroja MyBeem, ktoré vedú k rozšíreniu jeho funkcionality a jeho optimalizácii v súvislosti s analyzovanou problematikou. Praktickým výstupom práce je metóda agregácie záznamov o IP tokoch. Táto metóda používa pre zníženie záťaže monitorovacieho systému špeciálny spôsob agregácie založenej na redukcii kľúčových hodnôt toku.

Kľúčové slová

Agregácia, SLAmeter, monitorovanie, monitorovací systém, redukcia dát, MyBeem

Abstrakt v AJ

This thesis concern with reduction of IP flow information in order to decrease the SLAmeter monitoring system load. Implementation of these techniques was done in the MyBeem tool. The introduction of this thesis covers the analysis of SLAmeter tool, which uses MyBeem tool to meter various network characteristics. The analytic part also covers the techniques for IP flow information data reduction, mainly various techniques for data aggregation. In this thesis are designed, implemented and verificated the modifications of MyBeem tool which lead to functionality extensions and optimalization regarding to analysed issues. The practical output of this thesis is IP flow information data reduction technique. This method uses IP flow key values reduction as a special process to decrease the monitoring system load.

Kľúčové slová v AJ

Aggregation, SLAmeter, monitoring, monitoring system, data reduction, MyBeem

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **9.2.1 Informatika**

Študijný program: **Informatika**

Názov práce:

Redukcia informácií o IP tokoch za účelom zníženia záťaže monitorovacích systémov

Reduction of IP Flow Information in Order to Decrease the Monitoring Systems Load

Študent:	**Bc. Samuel Tremko**
Školiteľ:	**Ing. Peter Feciľak, PhD.**
Školiace pracovisko:	**Katedra počítačov a informatiky**
Konzultant práce:	**Ing. Adrián Pekár**
Pracovisko konzultanta:	**Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:

1. Analyzovať podmienky potreby redukcie informácií o IP tokoch v počítačových sieťach.
2. Navrhnúť úpravy nástroja MyBeem pre potreby redukcie informácií o IP tokoch.
3. Implementovať navrhnuté riešenie.
4. Overiť funkčnosť implementovaného riešenia.
5. Vypracovať dokumentáciu podľa pokynov vedúceho.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 02.05.2014

Dátum zadania diplomovej práce: 31.10.2013

.....
doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -11-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice 2. 5. 2014 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval vedúcemu mojej diplomovej práce Ing. Petrovi Feciľakovi, PhD., a najmä môjmu konzultantovi Ing. Adriánovi Pekárovi za cenné rady, odbornú pomoc, motiváciu a priaznivé pracovné prostredie pre tvorbu tejto diplomovej práce.

Taktiež moje poďakovanie patrí aj všetkým členom výskumnej skupiny MONICA.

Napokon by som rád poďakoval aj mojej rodine a priateľom za pomoc, trpezlivosť a pozitívnu motiváciu počas celého môjho vysokoškolského štúdia.

Predhovor

V oblasti dnešných konvergovaných počítačových sietí je kladený stále väčší dôraz na kvalitu služieb poskytovaných v týchto sieťach. Pre tento účel je výskumnou skupinou MONICA (Monitoring and Optimization of Network Infrastructures Communications and Applications) pôsobiacou v Laboratóriu počítačových sietí na Technickej univerzite v Košiciach vyvíjaný nástroj na pasívne monitorovanie a analýzu prevádzky počítačových sietí s názvom SLAmeter. Tento nástroj je samozrejme možné použiť aj na monitorovanie parametrov vypovedajúcich o kvalite poskytovaných služieb. Preto je nevyhnutné, aby nástroj fungoval spoľahlivo a dokázal poskytovať dôveryhodné informácie aj v kritických situáciách, ktoré sa v sieti môžu vyskytnúť. Táto diplomová práca vznikla z potreby riešiť kritický problém preťaženia nástroja MyBeem a zároveň aj celej architektúry, ktorý môže spôsobiť nespoľahlivosť a nefunkčnosť celého nástroja.

Tému Redukcia informácií o IP tokoch za účelom zníženia záťaže monitorovacích systémov som si zvolil preto, lebo sa zaujímam o problematiku počítačových sietí a taktiež aj o programovanie a realizácia myšlienky tejto diplomovej práce bola pre mňa výzvou. K výberu témy ma taktiež viedla niekoľkoročná skúsenosť s vývojom nástroja SLAmeter. Cieľom tejto práce je implementovať techniku na redukciu informácií o IP tokoch, ktorou je správne navrhnutá a implementovaná technika agregácie záznamov o IP tokoch. Tieto úpravy je nutné vykonať z dôvodu nasaditeľnosti nástroja SLAmeter do reálnej sieťovej prevádzky, aby tak bola zabezpečená jeho plná funkcionalita aj v náročných podmienkach, ktoré sa v počítačovej sieti môžu objaviť.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov a skratiek

ACP Analyzer Collector Protocol

atď. a tak ďalej

MyBeem BasicMeter Exporting and Measuring process

CNL Computer Networks Laboratory

CPU Central Processing Unit

ENIP (Ethernet/IP) Ethernet Industrial Protocol

ID Identificator

IE Information Element

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

IP Internet Protocol

IPFIX IP Flow Information eXport

JXColl Java XML Collector

MONICA Monitoring and Optimization of Network Infrastructures Communications and Applications

ms milisekunda

ns nanosekunda

OWD One Way Delay

QoS Quality of Service

resp. respektíve

RFC Request For Comments

SLA Service Level Agreement

TCP Transmission Control Protocol

tzv. takzvaný

UDP User Datagram Protocol

XML eXtensible Markup Language

Slovník termínov

QoS alebo aj Quality of Service je mechanizmus, prostredníctvom ktorého je možné regulovať kvalitu poskytovaných služieb, a to spôsobom nastavenia rôznych priorít jednotlivým užívateľom alebo aplikáciám.

MyBeem BasicMeter Exporting and Measuring process je najnižšia časť architektúry SLAmeter, ktorá slúži na monitorovanie údajov v počítačovej sieti a ich exportovanie zhromažďovaciemu procesu.

SLAmeter Service Level Agreement meter je nástroj na monitorovanie a anlýzu sieťovej prevádzky vyvíjaný výskumnou skupinou MONICA určený pre koncového používateľa.

IPFIX alebo aj IP Flow Information Export Protocol slúži na export informácií o IP tokoch v počítačových sieťach a zároveň je to názov pracovnej skupiny, ktorá tento protokol vyvíja[2].

RFC Request for Comments sú dokumenty publikované organizáciou IETF, ktoré popisujú metódy, výskum a inovácie aplikovateľné na systémy s napojením na internet.

flowId je identifikátor toku, ktorý je jedinečný v rámci pozorovacej domény [6].

Exportovací proces exportuje záznamy o tokoch jednému alebo viacerým zhromažďovacím procesom [2].

Merací proces jeho úlohou je vytváranie záznamov o tokoch. Okrem odchytávania paketov vykonáva aj funkcie filtrácie, značkovania a klasifikácie paketov, udržiava záznamy o tokoch a taktiež v ňom prebieha proces agregácie záznamov o tokoch [2].

Zhromažďovací proces prijíma záznamy o tokoch od jedného alebo viacerých exportovacích procesov [2].

Exportér je zariadenie nainštalované na meracom bode, ktoré obsahuje jeden alebo viacero exportovacích procesov [6].

Analyzér zariadenie, ktoré vyhodnocuje dáta po ich obdržaní z databázy alebo od protokolu ACP [6].

IANA je organizácia spravujúca globálne jedinečné čísla a mená v prostredí počítačových sietí.

Kolektor (zhromažďovač) je zariadenie obsahujúce aspoň jeden zhromažďovací proces [6].

Wireshark je voľne dostupný nástroj, ktorý dokáže odchytávať a analyzovať sieťovú prevádzku.

Proces agregácie proces redukcie informácií o IP tokoch [5].

Vzorkovanie paketov filtrácia paketov na základe stanovených kritérii [29].

Úvod

Prudký rozvoj výpočtovej techniky a počítačových sietí, ktorý sme zaznamenali v posledných desaťročiach prináša so sebou potrebu monitorovania a analýzy sieťovej prevádzky. Účelom monitorovania je kontorla poskytovania QoS, detekcia rôznych typov anomálií a útokov na počítačovú sieť, spojená s včasnou reakciou na ne a taktiež schopnosť predvídať rôzne scenáre, ktoré pri istých hodnotách parametrov prevádzky môžu v počítačovej sieti nastať.

Za týmto účelom vznikla na Fakulte elektrotechniky a informatiky Technickej univerzity v Košiciach výskumná skupina MONICA zaoberajúca sa vývojom programu na monitorovanie sieťovej prevádzky s názvom SLAmeter. Tento nástroj je vyvíjaný v konformite s protokolom IPFIX, a teda je možné kombinovať jeho jednotlivé časti s inými nástrojmi na monitorovanie počítačovej siete, ktoré boli tiež vytvorené v konformite s daným protokolom. Aplikácia SLAmeter má slúžiť bežnému používateľovi alebo aj poskytovateľovi internetového pripojenia na overenie a detekciu vyššie spomínaných udalostí.

Pri rastúcej výkonnosti počítačových sietí je potrebné v neposlednom rade poukázať na značné vyťaženie technických prostriedkov monitorovacej aplikácie pri nadmernom množstve monitorovanej prevádzky. Práve touto oblasťou problému sa zaoberá táto diplomová práca, keď sa snaží aplikovať mechanizmus na redukciu záťaže technických prostriedkov monitorovacej aplikácie pri čo najmenšej možnej strate granularity nameraných údajov, alebo strate údajov v celkovej miere, ktorá môže nastať v dôsledku neschopnosti aplikácie spracovávať nadmerné množstvo zachytených tokov z dôvodu preťaženia systému.

Samotná diplomová práca je členená do niekoľkých kapitol. Prvá kapitola obsahuje formuláciu úlohy diplomovej práce. V analýze sa postupne zaoberá problematikou a architektúrou SLAmetra a najmä jeho meracím a exportovacím nástrojom MyBeem, ktorého modifikácii sa venuje táto diplomová práca. Ďalej sa analýza zaoberá súčasnými spôsobmi monitorovania, technikami a algoritmami na obmedzenie vyťaženia

systémových zdrojov monitorovacej architektúry. V štvrtej kapitole sa práca zaoberá identifikáciou rozsiahlych tokov a detailnejšou charakteristikou niekoľkých parametrov sieťovej prevádzky, smerodajných pre efektívnosť monitorovania. Piata kapitola sa zaoberá návrhom modulu pre zníženie záťaže monitorvacieho systému SLAmeter. To znamená stanovenie podmienok, pri ktorých bude dochádzať k spusteniu redukčných metód, ďalej návrh samotného procesu redukcie a ostatných procesov potrebných pre jeho správne fungovanie. Šiesta kapitola sa zaoberá detailnejším popisom implementovaných zmien v nástroji MyBeem. V siedmej kapitole je overená správnosť navrhnutých a implementovaných úprav. Záverečná kapitola sa venuje zhodnoteniu dosiahnutých výsledkov celej diplomovej práce.

1 Formulácia úlohy

Cieľom tejto diplomovej práce je navrhnúť a implementovať metódu redukcie informácií o IP tokoch do procesu nástroja SLAmeter. Pod pojmom *redukcia informácií* sa v tejto práci chápe implementácia techniky, ktorá dokáže vhodným spôsobom reagovať na dianie v počítačovej sieti a na jeho základe upravovať spôsob spracovania informácií, aby sa tak predišlo preťaženiu a následnému zrúteniu celého monitorovacieho procesu. Implementovanie zmien sa týka najmä najnižšej časti tohto nástroja, monitorovacej aplikácie MyBeem, ktorá slúži ako sonda v počítačovej sieti. Ak je správne nainštalovaná na meracom bode, dokáže zachytávať informácie z počítačovej siete, spracovať ich a zaslať na analýzu ďalším prvkom architektúry. Všetky navrhnuté zmeny je nutné implementovať v konformite s protokolom a architektúrou IPFIX.

Je teda potrebné:

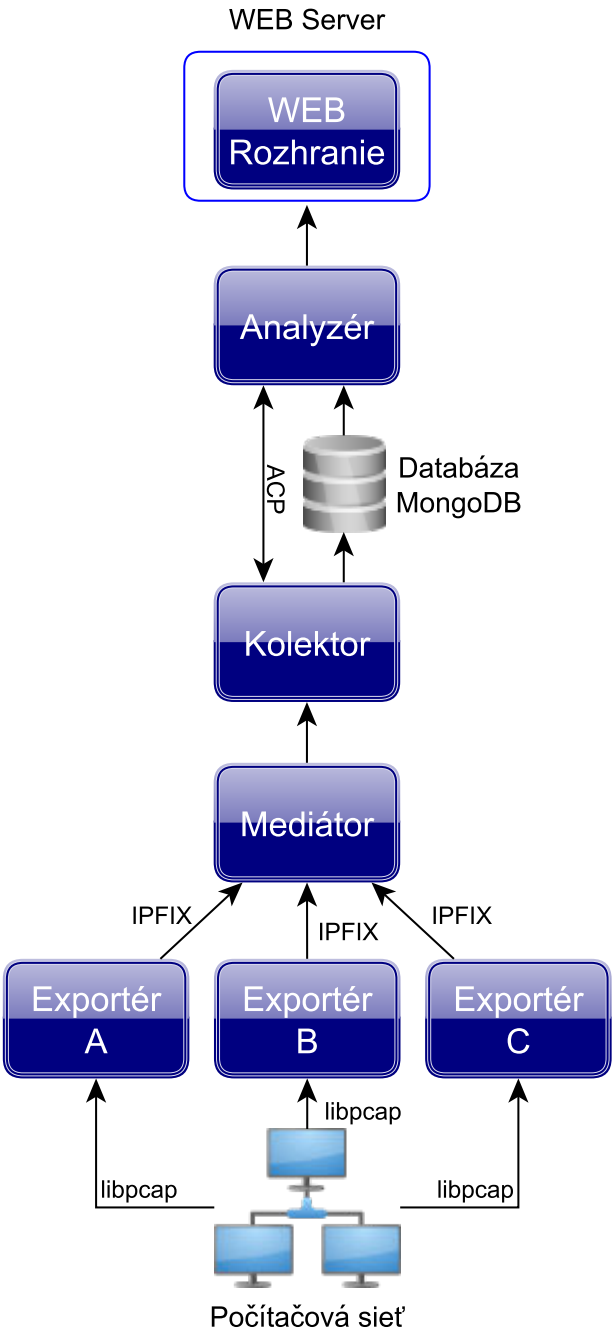
- bližšie popísať architektúru nástroja SLAmeter a nástroja MyBeem,
- definovať základné pojmy súvisiace s monitorovaním sieťovej prevádzky,
- analyzovať súčasné spôsoby redukcie množstva spracovávaných údajov,
- analyzovať príslušné techniky a algoritmy pre väčšiu efektivitu práce monitorovacieho procesu,
- analyzovať základné parametre sieťovej prevádzky, ktoré môžu byť smerodajné pre efektívnosť monitorovania,
- analyzovať súčasný stav redukcie spracovávaných záznamov v aplikácii MyBeem,
- navrhnúť spôsob redukcie množstva spracovávaných údajov aplikáciou MyBeem na základe analyzovaných skutočností, ktorá čo najmenším možným spôsobom ovplyvní detailnosť poskytovanej informácie vyšším prvkom architektúry SLAmeter, ale zvýši spoľahlivosť monitorovania

- navrhnúť štruktúru konkrétnych zmien v nástroji MyBeem, poprípade v celom nástroji SLAmeter,
- zhodnotiť možnosť použitia techniky vzorkovania paketov pre účely identifikácie rozsiahlych tokov,
- implementovať navrhnuté zmeny do procesu nástroja MyBeem, prípadne aj do iných komponentov nástroja SLAmeter,
- overiť funkčnosť implementovaných zmien a sledovať reakcie nástroja po nasadení do reálnej sieťovej prevádzky, aj prípadným vygenerovaním kritických situácií, v ktorých by monitorovacia aplikácia mohla zaznamenať určité problémy.

2 Analýza nástroja SLAmeter

Aj napriek tomu, že sa táto diplomová práca zaoberá najmä najnižšou vrstvou celého nástroja SLAmeter, exportérom IPFIX správ, je potrebné do jej analýzy zahrnúť aj samotnú architektúru nástroja SLAmeter ako celku. SLAmeter je jedným z projektov, ktorými sa zaoberá výskumná skupina MONICA pôsobiaca na Technickej univerzite v Košiciach pri laboratóriu počítačových sietí (CNL). Jeho hlavnou úlohou je monitorovanie sieťovej prevádzky, čo znamená proces od zachytávania údajov z počítačovej siete, cez ich prípadnú anonimizáciu, následne ich zhromažďovanie, prípadné uloženie v databáze alebo reálnočasové preposielanie Analyzéru, ich následné vyhodnocovanie a grafické zobrazenie na najvyššej vrstve nástroja, ktorou je webové rozhranie [23]. Tento celý proces je realizovaný buď v medziach charakteristík reálneho času, vtedy sa jedná o reálnočasové vyhodnocovanie, alebo ako následné vyhodnocovanie údajov po ich uložení v databáze. Celé monitorovanie a vyhodnocovanie je potrebné pre kontrolu QoS (Quality of Service) parametrov sieťovej prevádzky a dodržiavania zmluvy SLA (Service Level Agreement) o pokytovaní služieb, za ktoré je považované pripojenie do siete internet. Architektúra nástroja SLAmeter pozostáva z týchto častí:

- **Exportér** - sonda v počítačovej sieti, reprezentuje ho monitorovací a exportovací nástroj MyBeem
- **Mediátor** - je "nepovinným" prvkom architektúry, prebieha v ňom modifikácia IPFIX správ pred ich zhromaždením v Kolektore
- **Kolektor** - zhromažďovač IPFIX správ prijatých od exportéra alebo mediátora
- **Databáza** - databáza slúžiaca na uloženie nameraných údajov pre ich neskoršie spracovanie
- **Analyzér** - vyhodnocovač nameraných údajov získaných buď v reálnom čase



Obrázok 2 – 1: Architektúra nástroja SLAmeter

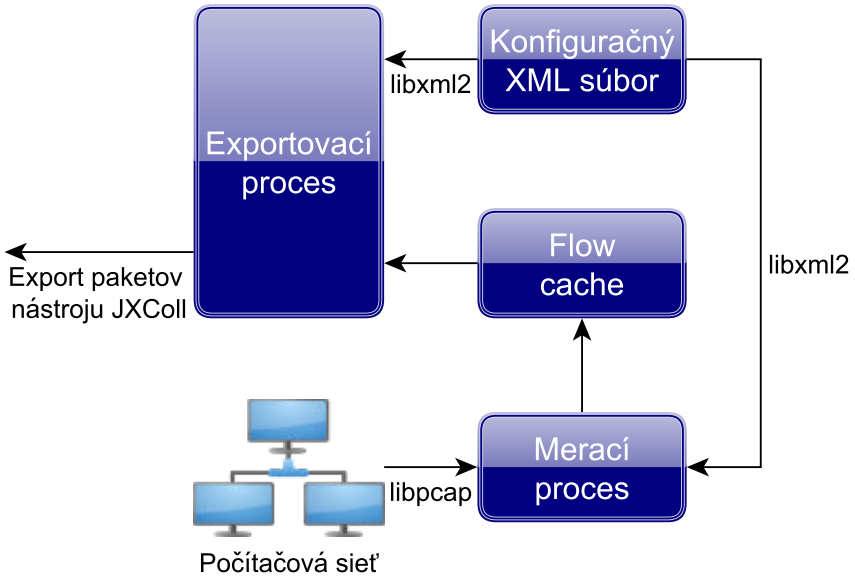
alebo selektovaných z databázy

- **WEB Rozhranie** - rozhranie slúžiace na grafické znázornenie vyhodnotených

údajov získaných od Analyzéra

Exportér funguje ako sonda v počítačovej sieti. Je nainštalovaný na meracom bode. Za **merací bod** pre nástroj SLAmeter sa dá považovať každý výpočtový systém, ktorý má funkčné sieťové rozhranie, operačný systém postavený na UNIX-ovom jadre a má nainštalovaný exportér, teda merací nástroj MyBeem, ktorý je potrebné príslušne nakonfigurovať, aby bol možný export informácií Kolektoru [14]. Prijaté informácie Kolektor buď uloží do databázy alebo ich v reálnom čase (v prípade, že je potrebné použit reálnočasové vyhodnocovanie) prepošle Analyzéru na vyhodnotenie. V Analyzéri sa prijaté dáta podľa požiadaviek WEB Rozhrania spracujú jednotlivými komponentami a odošlú sa na WEB Server, kde sú prostredníctvom WEB Rozhrania graficky zobrazené ako internetová stránka.

2.1 Analýza nástroja MyBeem



Obrázok 2 – 2: Architektúra nástroja MyBeem

Keďže táto diplomová práca sa primárne venuje popisu a implementácii zmien tejto

časti SLAmetra, je potrebné MyBeem bližšie charakterizovať. MyBeem je aplikácia napísaná v jazyku C, ktorá dokáže pracovať na operačných systémoch postavených na UNIX-ovom jadre. V aplikácii pracuje niekoľko vlákien, ktoré spracovávajú informácie. Ako môžme vidieť na obrázku 2 – 2, MyBeem pozostáva z dvoch procesov:

Merací proces - primárnou úlohou tohto procesu je odchytávanie paketov zo siete prostredníctvom *pcap* knižnice. Výsledkom tohto procesu je teda odchytávanie hlavičiek IP paketov, priradenie časových známok paketom, ich následné zaraďovanie do tokov, generovanie záznamov a tokoch, ich vedenie a ich ukladanie do vyrovnávacej pamäte tokov [24]. Vedením záznamov o tokoch rozumieme ich vytváranie, aktualizáciu, vytváranie štatistík a posúvanie týchto informácií exportovaciemu procesu (obr. 2 – 3). Preto musí existovať pre každý pozorovací bod zúčastnený na odchytávaní paketov zo siete minimálne jeden merací proces. Každý paket prechádzajúci pozorovacím bodom je preto spracovaný každým z meracích procesov, ktoré sú v danom okamihu na tomto pozorovacom bode aktívne.

Exportovací proces - jeho úlohou je zasielať informácie o IP tokoch uložených vo vyrovnávacej pamäti tokov Kolektoru prostredníctvom IPFIX správ [24].
IPFIX štandard presne definuje iba štruktúru týchto dvoch procesov, preto je možné medzi tieto dva procesy implementovať rôzne metódy agregácie a selekcie tokov, ktoré sú rozpracované v nasledujúcej sekcii tejto diplomovej práce.

2.2 Export IPFIX správ nástrojom MyBeem

Pretože táto diplomová práca v značnej miere pracuje s prvkami meracieho a exportovacieho procesu nástroja MyBeem, ako aj s pojmami *uniflow* a *biflow*, taktiež s pojmami *aktívna expirácia* a *pasívna expirácia* a s nimi súvisiacimi pojmami *active timeout* a *passive timeout*, je potrebné tieto pojmy bližšie charakterizovať a uviesť ich spojitosť s nástrojom MyBeem.

Obrázok 2 – 3: Architektúra meracieho procesu

K exportu toku v procese jeho spracovania v nástroji MyBeem dochádza v prípade, že nastala jeho expirácia. V danej situácii rozlišujeme tri druhy expirácie toku:

- **aktívna expirácia** - ak do daného toku prichádzajú pakety po dobu stanovenú v ukazovateli *active timeout* a nedošlo pri tom k pasívnej expirácii toku, tok expiroval aktívne. V takomto prípade je kópia záznamu o toku exportovaná zhromažďovaciemu procesu, a ak do daného toku príde ďalší paket, pôvodné hodnoty, nachádzajúce sa pre daný tok vo vyrovnávacej pamäti tokov, sú opäť aktualizované. Ak však do toku žiaden ďalší paket nepríde, tok expiruje pasívne a je exportovaný.
- **pasívna expirácia** - ak do daného toku neprichádzajú žiadne pakety po dobu

passive timeout, tak tok expiroval pasívne a je exportovaný zhromažďovaciemu procesu. V prípade exportu po pasívnej expirácii záznam o toku vo vyrovnávacej pamäti tokov nástroja MyBeem zaniká. Ak merací proces zachytí ďalší paket prislúchajúci danému pasívne expirovanému a exportovanému toku, je pre neho vytvorený nový záznam vo vyrovnávacej pamäti tokov.

- **expirácia toku spôsobená zachytením koncového paketu toku** - ak merací proces zachytí koncový paket toku (FIN paket), znamená to, že tok je ukončený a teda je možné ho presunúť do pamäte expirovaných tokov a exportovať. Po exporte toku zaniká záznam o tomto toku vo vyrovnávacej pamäti tokov.

Časové údaje *active timeout* a *passive timeout* je možné manuálne nakonfigurovať prostredníctvom konfiguračného súboru *config.xml* nástroja MyBeem. Ešte pred samotným exportom tokov sú toky vkladané do pamäte expirovaných tokov a až následne sú zaslané zhromažďovaciemu procesu exportovacím vláknom.
Nástroj MyBeem je možné nakonfigurovať v dvoch režimoch práce prostredníctvom nastavenia v konfiguračnom súbore programu. Tieto režimi sú:

- **uniflow** - pri identifikácii príslušnosti paketov voči tokom je nutné, aby sa hodnoty polí paketu zhodovali s odpovedajúcimi kľúčovými hodnotami toku, ktorého záznam sa nachádza vo vyrovnávacej pamäti tokov. To znamená, že pre každý smer komunikácie vzniká samostatný záznam o toku.
- **biflow** - pri tejto konfigurácii funguje rovnaký proces, ako pri *uniflow* konfigurácii, avšak naviac sú kontrolované aj odpovedajúce kľúčové hodnoty záznamu o toku v opačnom smere. Teda pri tomto type konfigurácie vzniká jeden spoločný záznam o toku pre oba smery komunikácie.

Pred zaslaním prvého záznamu o toku zhromažďovaciemu procesu MyBeem zašle šablónu, v ktorej sú okrem jej identifikátora definované informácie o počte a rozsahu informačných elementov, ktoré budú zasielané v IPFIX správach prislúchajúcich k

tejto šablóne. V ďalšom procese odosielania IPFIX správ zhromažďovaciemu procesu je šablóna posielaná v pravidelných intervaloch, ktoré sú určené nastavením položky *refreshTemplateTime* v konfiguračnom súbore programu. Zároveň každá IPFIX správa obsahuje identifikátor šablóny, aby zhromažďovací proces vedel, ktorú šablónu použiť na dekódovanie informácie obsiahnutej v príslušnej IPFIX správe.

3 Analýza potreby zníženia záťaže monitorovacích systémov

Pojem zníženia záťaže monitorovacích systémov úzko súvisí s pojmom optimalizácie monitorovania sieťovej prevádzky, keďže práve táto metóda exportovania údajov z počítačovej siete má dopomôcť k zvýšeniu spoľahlivosti monitorovania. Pojem optimalizácie monitorovania sieťovej prevádzky popisuje návrh a implementáciu mechanizmov a metód, ktoré zabezpečia odstránenie nedostatkov monitorovacieho procesu [12]. Preto by sa optimalizácia mala zaoberať najmä:

- zvýšením presnosti mechanizmov vyhodnocovania
- minimalizáciou dodatočného záťaženia siete spôsobeného jej monitorovaním
- zvýšením schopnosti vyhodnocovať namerané údaje v reálnom čase
- efektivitou využitia spojenou s minimalizáciou zaťaženia sieťových prostriedkov monitorovacími mechanizmami
- zefektívnením monitorovacieho procesu, za účelom zníženia záťaže monitorovacej aplikácie a ostatných prvkov architektúry, ktoré na ňu nadvezujú
- zvýšením spoľahlivosti monitorovacieho systému v takej miere, aby bol schopný spracovávať aj veľké množstvá záznamov o tokoch bez ich úplnej straty v dôsledku vyťaženia

Na dosiahnutie daných cieľov môžu v značnej miere poslúžiť techniky na redukciu informácií o IP tokoch a z nich najmä technika agregácie záznamov o IP tokoch, ktorej sa v podstatnej miere venuje táto diplomová práca.

3.1 Agregácia tokov

Na to, aby sme detailnejšie porozumeli problematike agregácie záznamov o IP tokoch, je potrebné zadefinovať niekoľko pojmov, ktoré s touto témou veľmi úzko súvisia.

3.1.1 Tok

Je definovaný ako súbor IP paketov, ktoré sú zachytené v danom časovom intervale konkrétnym meracím bodom v počítačovej sieti. Všetky pakety, ktoré patria do konkrétneho toku obsahujú súbor rovnakých parametrov, päticu ukazovateľov (IP-five-tuple [11]) - typ protokolu, zdrojovú a cieľovú IP adresu, zdrojový a cieľový port. Každý z určujúcich parametrov je definovaný ako výsledok aplikovania funkcie na hodnoty [2]:

- jedna alebo viacero polí v hlavičke paketu(cieľová IP adresa), polia transportného protokolu v hlavičke paketu (cieľový port), polia aplikačného protokolu v hlavičke paketu (položky hlavičky RTP protokolu [3])
- jedna alebo niekoľko charakteristík týkajúcich sa paketu ako takého (MPLS návestia)
- jedno alebo niekoľko polí, ktoré sú odvodené pri spracovaní paketu (IP adresa nasledujúceho uzla)

Samotný paket patrí do daného toku iba vtedy, ak spĺňa všetky vyššie definované podmienky.

3.1.2 Agregovaný tok

Je odvodením resp. zlúčením viacerých tokov podľa vopred definovaných agregačných podmienok v procese agregácie. Od momentu, kedy je agregovaný tok exportovaný, stáva sa klasickým záznamom o toku, ako je definované v [1] , a preto je potrebné s ním týmto spôsobom pracovať [5].
Rozlišovanie tokov na základe spomenutej pätice ukazovateľov má pri veľkom objeme prevádzky alebo pri rôznych typoch útokov za následok obrovské množstvo nameraných údajov, ktoré musia byť spracované a exportované meracím bodom a následne prijaté a spracované v analyzujúcej aplikácii. V snahe zredukovať spracovávané množstvo dát komerčné monitorovacie aplikácie exportujú iba záznamy o tokoch s veľkým objemom dát. Inou metódou je aplikácia vzorkovacích algoritmov, ktoré podobne ako pri vzorkovaní paketov, vyberú na základe daných kritérii iba podmnožinu záznamov o tokoch, ktorá bude exportovaná. Štúdia [11] uvádza, že ak zoberieme do úvahy granularitu nameraných údajov, tak zisťujeme, že množstvo aplikácií nevyžaduje detailnosť nameraných dát na úrovni spomínanej pätice ukazovateľov (príznačnou skupinou sú napríklad aplikácie účtovania slúžiace na zistenie objemu dát prenesených zákazníkom).

3.1.3 Technika agregácie tokov ako spôsob redukcie záznamov

Technikou agregácie tokov je možné efektívne zredukovať množstvo nameraných údajov, a to vyradením niektorých kľúčových hodnôt toku a jedho následným zlúčením s iným záznamom o toku, ktorý sa nachádza vo vyrovnávacej pamäti tokov a má s týmto tokom spoločné ostatné kľúčové parametre. Výsledkom je agregovaný záznam o toku, ktorý popisuje tú istú sieťovú prevádzku, avšak zahŕňa menej detailov.

- Bežnou metódou na redukciu množstva monitorovaných dát je redukcia časovej granularity toho istého toku, to znamená spojenie týchto záznamov do jedného pokrývajúceho väčší časový interval.

- V prípade aplikácii fungujúcich na báze modelu klient-server je informácia o požadovanej službe zo strany klienta zväčša obsiahnutá v čísle portu na strane servera. Číslo portu na strane klienta je často dynamicky pridelené operačným systémom klienta. Preto je možné vylúčiť klientské číslo portu zo záznamu bez rizika straty cenných informácií.
- Účtovacie aplikácie zväčša potrebujú iba informácie o množstve prevádzky, ktorá je prijatá alebo odoslaná klientom, a teda informácia o každom toku nieje podstatná. V takomto prípade je možné zo záznamu vylúčiť zdrojovú a cieľovú IP adresu komunikácie a taktiež čísla portov.

V súčasnosti sa rôzne techniky agregácie tokov uplatňujú najmä na strane analyzéra, kde sú podľa potrieb aplikácie monitorované dáta komprimované a nepotrebná informácia je zahodená [5]. Napriek všetkému, agregácia vykonávaná v analyzéri, alebo inom vyššom prvku meracej architektúry žiadnym spôsobom neredukuje množstvo nameraných dát generovaných meracím bodom a následne zasielaných vyšším vrstvám a napokon analyzéru. Týmto záťaž ostáva na vštkých častiach monitorovacej architektúry a prenáša sa postupne od najnižšej vrstvy až k tej najvyššej, kde je táto prebytočná informácia tak, či onak, zahodená.

3.1.4 Problémy pri monitorovaní nástrojom SLAmeter

V reálnej praxi, pri monitorovaní siete nástrojom SLAmeter, vzniká problém preťaženia najmä v databáze, ktorá slúži ako úložisko nameraných záznamov o tokoch resp. následne aj pri dotazovaní a samotnom vyhodnocovaní obrovského množstva údajov. Problém spočíva v tom, že databáza nedokáže efektívne obsluhovať obrovské množstvá údajov, ktoré sa v nej nazhromaždia počas merania. Jedná sa rádovo o státisíce tokov, pri ktorých výbere z databázy analyzujúcov aplikáciou, reagujúcou na požiadavku WEB rozhrania, dochádza k oneskoreniam rádovo aj v desiatkach sekúnd, čo je neprípustné pre efektívne analyzovanie nameraných hodnôt. Tento

problém by bolo možné do značnej miery riešiť agregáciou nameraných záznamov o tokoch, pretože takto by databáza neobsahovala záznamy aj pre zdanlivo bezvýznamné toky o veľkosti pár oktetov, a tak by sa významne znížil počet záznamov a s ním aj čas, ktorý je potrebný na výber údajov z databázy analyzujúcou aplikáciou. Obrázok 3 – 1 zobrazuje neagregovaný záznam o tokoch. Ak aplikujeme na tieto

Prot	SrcPort	SrcAddr	DstPort	DstAddr	#Pkt	#Oct	Start	End
TCP	64235	10.0.1.1	80	10.10.0.10	***4***	***144***	***1055***	1090
TCP	64236	10.0.1.1	80	10.10.0.10	***3***	***56***	1071	***1103***
TCP	6889	10.0.1.2	80	10.10.0.10	***2***	***34***	1083	1100
TCP	5555	10.0.2.1	80	10.10.0.10	6	155	1090	1201
TCP	6666	10.0.2.1	80	10.10.0.11	3	77	1095	1199

Obrázok 3 – 1: Tabuľka zachytených tokov pred použitím agregácie [11]

SrcAddr	DstAddr	#Pkt	#Oct	Start	End
10.0.1.0/24	10.10.0.10	9	234	1055	1103
10.0.2.0/24	10.10.0.10	6	155	1090	1201
10.0.2.0/24	10.10.0.11	3	77	1095	1199

Obrázok 3 – 2: Tabuľka zachytených tokov po použití agregácie [11]

údaje agregačné pravidlo, v ktorom:

- ak sa jedná o protokol TCP, tento indetifikátor zahodíme
- vylúčime zdrojový port a taktiež cieľový port (ak sa jedná o cieľový port 80 alebo 443)
- zlúčime záznam o zdrojovej IP adrese do prefixu /24
- sčítame počty paketov a ukazovateľ octetTotalCount

- za začiatok toku zvolíme najnižšiu hodnotu parametra flowStartMilliSeconds a za koniec najvyššiu hodnotu parametra flowEndMilliSeconds

Takto dostávame jeden agregovaný záznam o toku pozostávajúci z troch originálnych záznamov, ktorý je vyznačený na obrázku 3 – 2 [11].

3.1.5 Metóda *"Gradual Flow Key reduction"*

V štúdii [7] autori navrhujú systém agregácie záznamov *"Gradual Flow Key reduction method"*, ktorý spočíva v postupnej redukcii kľúčových hodnôt toku podľa priority tak, ako je to znázornené v tabuľke 3 – 1. Pričom to, kedy sa začne daná redukcia vykonávať je vopred stanovené používateľom systému udaním tzv. *kritickej hodnoty* počtu záznamov v pamäti.

Záznamy o tokoch, ktoré sú obdržané od meracieho procesu su usporiadané do

Tabuľka 3 – 1: Priorita jednotlivých kľúčových hodnôt toku [7]

Informačné elementy slúžiace ako kľúčové hodnoty	Priorita kľúčových hodnôt
protocolIdentifier	1 (najvyššia)
destinationIPv4Address/destinationIPv6Address	2
sourceIPv4Address/sourceIPv6Address	3
destinationTransportPort	4
sourceTransportPort	5 (najnižšia)

dvoch zoznamov. Prvý zoznam je určený pre originálne a detailné záznamy, z ktorých sú vybrané tie, ktoré je možné agregovať, pretože pre používateľa neobsahujú informácie o dôležitých tokoch. Tieto záznamy sú zapísané do druhého zoznamu. Záznamy, ktoré je potrebné nechať v originálnej podobe (jedná sa o rozsiahle toky) ostávajú v prvom zozname. V druhom zozname sa postupne záznamy agregujú podľa kľúčových identifikátorov toku tak, že záznam s tzv. nižšou prioritou je zlúčený do

jedného väčšieho agregovaného záznamu so záznamom vyššej priority (teda záznam vyššej priority pokrýva ten nižší), avšak nedochádza k redukcii kľúčových hodnôt. Originálne záznamy sú po zlúčení a aktualizovaní položiek o toku zahodené.
Ak ani táto metóda neprinesie požadovanú redukciu záznamov o tokoch, je potrebné postupne zahadzovať jednotlivé kľúčové ukazovatele toku (prioritne podľa tabuľky 3 – 1) a takto dosiahnuť novú redukovanú *n-ticu* kľúčových hodnôt a teda aj možnosť výraznejšej agregácie, keďže dochádza k zníženiu granularity. Pre každú skupinu redukovaných tokov sa vytvára samostatný zoznam, v ktorom sú umiestnené len toky, ktoré majú redukované tie isté kľúčove parametre. Tento proces redukcie pokračuje, až kým sa nedosiahne požadovaný výsledok zníženia počtu záznamov. V zásade platí, že určovanie priority jednotlivých kľúčových hodnôt toku záleží na rozhodnutí používateľa, teda používateľ stanoví, v akom poradí budú jednotlivé kľúčové hodnoty redukované.
Tabuľka 3 – 2 demonštruje funkcionalitu popísanej metódy *"Gradual Flow Key reduction method"*. Je tu znázornených niekoľko tokov, ktoré sa v danom čase vyskytujú vo vyrovnávacej pamäti programu. Dva z týchto tokov sú agregované o jeden alebo viacero kľúčových hodnôt a jeden tok je ponechaný ako neagregovaný z dôvodu jeho rozsahu, a teda jeho vyššej priority. Ukazovateľ počtu agregovaných tokov *OriginalFlowsPresent* znázorňuje počet originálnych tokov agregovaných v danom zázname. Ak je hodnota tohto ukazovateľa rovná jednej, znamená to, že daný záznam obsahuje jeden tok, a tým je on sám, teda pôvodný neagregovaný záznam o toku.
Autori štúdie [7], na rozdiel od autorov štúdie [11], ktorý napríklad v prípade zdrojovej IP adresy túto adresu nezahadzujú, ale pristupujú len k jej zovšeobecneniu na základe masky podsiete, autory [7] navrhujú postupnú možnosť redukcie kľúčových hodnôt toku na úroveň zahodenia všetkých kľúčových hodnôt, čo spôsobí vytvorenie jedného všeobecného záznamu o toku v poslednom z bufferov, do ktorého bude možné zaradiť každý, ešte predtým nezaradený tok. Samozrejme sa predpokladá aktualizácia všetkých atribútov, ktoré obsahujú informácie o danom toku pri zlúčení

Tabuľka 3 – 2: Metóda Gradual Flow Key reduction [7]

Prot.	SrcIP	DstIP	SrcPort	DstPort	OrigFlowsPrs
UDP		192.168.0.123	4003	832	32
TCP	132.16.34.234	86.35.232.64	2309	3208	1
UDP				2400	140

každej dvojice záznamov o tokoch do jedného.

Pre implementáciu v programe MyBeem je pre nás podstatný výsledok tejto agregačnej techniky, ktorý dosiahneme redukciou kľúčových hodnôt toku, a tým je predpokladané zníženie záťaže vyrovnávacej pamäte monitorovacieho a exportovacieho nástroja MyBeem a taktiež zníženie zaťaženia vyšších prvkov architektúry.

3.2 Vzorkovanie

Vzorkovanie paketov (angl. packet sampling) je jednou z techník redukcie záťaže pamäte v meracom procese a redukcie množstva spracovávaných záznamov. Hlavnou myšlienkov tejto techniky je výber iba určitej podmnožiny paketov (*Selection Process* [30]) prechádzajúcich meracím bodom, ktoré budú v ďalšom procese spracovávané [29]. Vo všeobecnosti štandard definuje trojicu možných druhov vzorkovania paketov:

- **Náhodná selekcia** *(Random Selection)* - z paketov toku je paket pre ďalšiu analýzu vyberaný náhodne, bez aplikovania akéhokoľvek typu filtra resp. kritéria.
- **Deterministická selekcia** *(Deterministic Selection)*- je technikou filtrácie paketov toku na základe ich obsahu. Z paketov toku je stále vyberaný každý n-tý paket, ktorý spĺňa vopred definované obsahové kritériá.
- **Deterministická aproximácia** *(Hash-based Selection)* - je podobne ako *de-*

deterministická selekcia technikou filtrácie paketov toku na základe ich obsahu. Na obsah paketu je aplikovaná hash funkcia. Paket je vybraný, ak výsledok funkcie spadá do určitého intervalu hodnôt.

Technika vzorkovania paketov má vplyv na presnosť nameraných údajov, keďže je stále vyberaný iba určitý počet paketov toku a teda stratovosť údajov je vyššia ako pri implementácii agregácie tokov. Je vhodná najmä pri dlhodobom monitorovaní IP tokov aplikáciami, ktoré nevyžadujú prílišnú presnosť a granularitu nameraných údajov.
Vhodnejšou technikou znižovania záťaže pri potrebe detailnej analýzy každého paketu toku je agregácia záznamov, pretože pri tejto technike nedochádza k strate údajov spôsobenou selekciou n paketov toku. Vzorkovanie paketov je samozrejme veľmi rozšírenou a používanou metódou, ktorá je taktiež implementovaná v nástroji MyBeem a jej myšlienka a užitočnosť nemôžu byť spochybňované.

4 Identifikácia rozsiahlych tokov

Ako je v súčasnosti známe, aplikácie, ktoré sa zaoberajú monitorovaním sieťovej prevádzky pristupujú k zachyteným dátam ako k súboru IP tokov, ktoré je potrebné analyzovať. Avšak pri súčasných trendoch neustáleho zvyšovania priepustností a rýchlostí liniek, čo má za následok zvyšovanie počtu prenášaných IP tokov, je veľmi pamäťovo náročné udržiavať každý ukazovateľ o danom toku aktuálny. Merania ukázali, že malé množstvo veľkých tokov tzv. "heavy hitters", teda tokov, ktoré zaberajú viac ako 0,1% celkovej kapacity linky v danom časovom intervale, tvorí signifikantný podiel na prevádzke. Častokrát je problémom, práve kvôli množstvu malých tokov, o ktorých je potrebné viesť záznamy, tieto rozsiahle toky sledovať a aktualizovať s nimi spojené štruktúry v pamäti meracieho procesu, čím môžme prichádzať o cenné informácie, najmä čo sa týka rôznych metód účtovania. Preto, ako aj uvádza štúdia [10] je v mnohých situáciách potrebné tieto rozsiahle toky identifi-

kovať a vedieť sledovať množstvo údajov, ktoré pokrývajú, a zároveň uplatniť rôzne techniky selekcie na pakety, ktoré patria do "menej" podstaných malých a krátkych tokov, čím znížime zaťaženie pamäte.

Merania množstva prevádzky medzi jednotlivými AS a vo vnútry jednotlivých AS ukázali [26], že množstvo tokov medzi dvoma stanicami v priebehu jednej hodiny bolo zhruba na úrovni 1,7 milióna zachytených tokov. Po použití rôznych techník agregácie bolo toto číslo na úrovni 0,5 milióna tokov. Preto môže byť častokrát problémom pre pamäť zariadenia, na ktorom beží merací proces, a ktoré žiadnym spôsobom nevyužíva techniku agregácie, udržiavať záznamy o takomto množstve tokov. Ďalej štúdia [26] uvádza, že aj napriek obrovskému množstvu zachytených tokov, len **9% tokov** medzi jednotlivými AS pármi zodpovedá za **90% celkového objemu sieťových dát** medzi všetkými AS pármi.

Protokol CISCO NetFlow sa snaží tento problém odstrániť technikou *packet sampling*-u [27]. Nevýhodou tejto metódy je však to, že má vplyv na celkovú presnosť merania.

Identifikácia tzv. "heavy hitters" môže napomôcť pri úprave správania sa meracieho procesu v metóde redukcie informácií o IP tokoch napr. aplikáciou rôznych druhov politík na jednotlivé skupiny tokov, a tým zefektívniť jeho funkcionalitu aj pri zvyšujúcej sa záťaži. Štúdia [4] uvádza, že identifikácia týchto tokov môže výrazne dopomôcť pri optimalizácii fyzickej topológie siete a smerovacích politík.

4.1 Použiteľnosť implementácie vzorkovania v nástroji MyBeem pre potreby agregácie

V nástroji MyBeem je implementovaných niekoľko druhov vzorkovacích techník, ktoré selektujú pakety tokov na základe vopred stanovených charakterisík. Implementované sú techniky:

- ***Systematic count based sampling*** - táto technika stanovuje dva typy in-

tervalov, a to *samplingPacketInterval* a *samplingPacketSpace*. *SamplingPacketInterval* určuje počet paketov v rade, ktoré budú odchytené a podrobené ďalšej analýze. *SamplingPacketSpace* určuje počet paketov medzi jednotlivými selekciami.

- ***Systematic time based sampling*** - technika taktiež stanovuje dva typy intervalov, a to *samplingTimeInterval* a *samplingTimeSpace*. *SamplingTimeInterval* stanovuje časový interval, v ktorom sú pakety odchytávané. *SamplingTimeSpace* stanovuje časový interval medzi jednotlivými časovými intervalmi selekcie paketov.
- ***Random n of N sampling*** - pri tomto type vzorkovania je vyberaných n paketov z tzv. "rodičovskej populácie" N paketov na základe náhodného výberu.
- ***Uniform probability sampling*** - táto technika hovorí o tom, že každý element má rovnakú pravdepodobnosť výberu P z "rodičovskej populácie", určenú číslom s pohyblivou rádovou čiarkou typu *float*.

Rôzne štúdie sa zaoberajú možnosťou použitia techniky vzorkovania paketov nie ako selektívnej techniky, ktorá určuje, či daný paket bude zahodený, alebo bude podrobený ďalšej techike analýzy v procese spracovania, ale ako techniky pre označovanie veľkých a malých tokov. Ak je teda onačených n paketov toku a súčet hodnôt ukazovateľov *octetTotalCount* je väčší ako stanovená podmienka, daný tok bude považovaný za rozsiahly a teda nebude agregovaný. Problém môže nastať práve v spôsobe označovania, ktoré je buď náhodne stanovené, alebo je nastavený interval selekcie, čo môže spôsobiť značné nepresnosti v identifikácii. Bude preto potrebné podrobiť techniku vzorkovania paketov analýze a tak určiť jej spoľahlivosť v porovnaní s navrhovaným spôsobom určovania rozsiahlych tokov v tejto práci.

4.2 Analýza súčasného stavu spôsobov merania

Z doterajšej praxe v oblasti monitorovania sieťovej prevádzky je možné vydedukovať, že meracie mechanizmy, v dôsledku meniaceho sa charakteru sieťovej prevádzky, často neefektívne využívajú systémové prostriedky, čo znamená, že sa neprispôsobujú jej aktuálnemu stavu a záťaži systémových zdrojov. Výrazným nedostatkom v danej oblasti je absencia adaptívnych exportovacích metód, ktoré by zohľadňovali vyššie spomenuté problémy [8]. Súčasné meracie a exportovacie mechanizmy zápasia pri preťažení nadmernou prevádzkou alebo inými situáciami v sieti s problémami:

- nadmerná prevádzka má za následok prečerpanie dostupnej vyrovnávacej pamäte tokov a následné zahadzovanie nových legitímnych zachytených tokov
- export ešte neukončených tokov so zámerom uvoľnenia priestoru pre nové toky, čo môže mať za následok preťaženie exportovacích kapacít a v konečnom dôsledku aj preťaženie kolektora
- aplikovanie vzorkovania podľa úrovne prevádzky, čo môže mať za následok zníženie presnosti účtovania a stratu možno dôležitých údajov o tokoch

V štúdii [9] autory poukazujú najmä na hlavnú požiadavku pri súčasnom monitorovaní - zvýšenie presnosti merania pri znižujúcich sa nákladoch na dané meranie. Zároveň však poukazujú na problém, že je veľmi obtiažne nastaviť vzorkovanie tak, aby bola zachovaná dostatočná presnosť a zároveň nedošlo k preťaženiu meracích prostriedkov v žiadnej z fáz merania. Zameraním sa na Cisco proprietárny protokol NetFlow [28] navrhujú implementáciu dvoch samostatných, kvázi adaptívnych algoritmov:

- **CPU sampling process** - tento proces dohliada na množstvo paketov spracovaných procesorom a zároveň kontorluje množstvo dostupných prostriedkov CPU. Ak by malo dôjsť k preťaženiu procesora pri náraste prevádzky v sieti, ktorú je potrebné odchytiť a spracovať, proces zareaguje zmenou granularity

vzorkovania na nízku hodnotu. Opačná zmena nastáva pri znížení hustoty prevádzky.

- **Memory sampling process** - na vzorkovanie paketov tokov je použitý *Sample and hold* algoritmus [10] pri súčasnej kontrole dostupných prostriedkov vo vyrovnávacej pamäti tokov. To znamená, že do vyrovnávacej pamäte tokov sú zaradené iba tie toky, ktoré sú považované za tzv. *heavy hitters*

Pre návrh algoritmov a metód na úpravu súčasného stavu exportu informácií o IP tokoch je potrebné zohľadniť niektoré z charakteristík siete. Totižto, v počítačovej sieti existuje niekoľko ukazovateľov, na základe ktorých by bolo možné v ďalšom procese implementácie redukčných metód tieto metódy upraviť do takej podoby, aby boli schopné adaptívne reagovať na situácie vyskytujúce sa v sieti, ba dokonca takéto situácie predvídať. Tieto procesy by bolo možné realizovať aj na základe výsledkov kombinácie hodnôt ukazovateľov analyzovaných v nasledujúcej sekcii a tak navrhnúť vhodné reakcie na dané krízové situácie.

4.3 Dôležité parametre sieťovej prevádzky

V oblasti monitorovania počítačových sietí môžme nájsť niekoľko parametrov sieťovej prevádzky, ktoré sú ovplyvnené samotným stavom siete. Zmena týchto parametrov môže byť vnímaná ako výsledok interakcie medzi prevádzkou a sieťovou infraštruktúrou [17]. Podrobnosti, ktoré so sebou tieto parametre prinášajú sú spojené s problematikou QoS.

Ak vnímame dynamiku prevádzky ako systém, ktorý sa na základe určitých charakteristík mení v čase (vyvíja sa), pričom tieto charakteristiky je možné použitím vhodných modelov (napr. Skryté Markovské Modely - SMM) odhadnúť ba dokonca i predvídať, je možné následne na základe týchto parametrov predvídať aj nasledujúci stav celého systému [16]. Preto by ideálnym spôsobom ako v budúcnosti upravovať funkcionalitu meracieho procesu a taktiež celý proces agregácie záznamov o tokoch

bolo vypracovanie adaptívnych predikčných metód, ktoré na základe analýzy aktuálnej sieťovej prevádzky, resp. na základe zvolených kľúčových ukazovateľov (tým sa nemyslí pätica kľúčových hodnôt toku) dokázali do istej miery predpovedať situáciu v sieti a ďalej modifikovať správanie sa meracieho nástroja a celého procesu agregácie. Tieto parametre sú rozpracované v nasledujúcich podkapitolách tejto diplomovej práce.

4.3.1 Priepustnosť (*throughput*)

Touto veličinou popisujeme mieru, ktorou je prevádzka schopná reálne "prúdiť" v sieti. Maximálna priepustnosť je všeobecne určená kombináciou kapacitných možností jednotlivých sieťových komponentov a nepriechodnosti, ktorá je zapríčinená prevádzkou v sieti. Ak je časový interval T veľký v porovnaní s časom potrebným na prenos po určitej trase v sieti, tak priepustnosť trasy T_h vieme určiť ako podiel [17]

$$T_h = C_n/T \tag{4.1}$$

kde sa premenná C_n vzťahuje na počet paketov, ktoré sú prenesené cez celú trasu bez straty počas časovej periódy n.

4.3.2 Šírka pásma (*bandwidth*)

Tento pojem sa podobne ako pojem *priepustnosť* používa na ohodnotenie sieťových komponentov a predstavuje množstvo údajov, ktoré je možné fyzicky preniesť daným médiom za jednotku času, pričom platí, že každý zo sieťových komponentov môže mať inú šírku pásma, prípadne rôzne kapacitné vlastnosti [17]. Je však potrebné rozlišovať pojem *priepustnosť* a *šírka pásma*.

Priepustnosť predstavuje množstvo informácií (bitov, bajtov, paketov, atď.), ktoré je *reálne* prenositeľné cez dané spojenie (reálna vlastnosť siete), kým pojem *šírka pásma* popisuje množstvo údajov, ktoré je možné $fyzicky$ preniesť daným médiom

(teoretická vlastnoť siete). Maximálna priepustnosť celkovej trasy spojenia (postupnosti liniek) od zdrojového uzla ku koncovému uzlu siete je určená linkou s najnižšou priepustnosťou. Táto linka sa tiež nazýva aj *linka úzkeho profilu (bottleneck link)*. Poznáme tri základné typy šírky pásma, ktoré je vhodné brať do úvahy z hľadiska optimalizácie výkonnsoti siete:

Kapacita - predstavuje maximálnu možnú priepustnosť, ktorú je schopná daná linka poňať. Niekedy je označovaná aj ako "priechodná" (uncongested) šírka pásma. Kapacita je vlastnosť trasy, ktorá sa mení, len ak sa mení rýchlosť smerovača alebo linky.

Dostupná šírka pásma - označuje sa aj ako *reziduálna* alebo *zvyšková* kapacita - je tá časť kapacity linky, ktorá nie je v danom časovom okamihu používaná. To znamená, že ak máme linku s kapacitou 100 Mb/s a aktuálne je používaných 40 Mb/s, potom dostupná šírka pásma je 60 Mb/s.

Kapacita hromadného prenosu - *bulk transfer capacity* predstavuje šírku pásma, ktorú získa nové dlhotrvajúce TCP spojenie v rámci danej cesty. Pre šírku pásma je totižto charakteristické to, že v prípade už existujúcich spojení ďalšie nové spojenie nemusí dostať celú dostupnú šírku pásma [17].

Často využívanou metódou na meranie kapacity je metóda *paketových párov (packet-pair method)* [25, 21] alebo metóda *veľkosti oneskorenia (size-delay method)* [19, 22]. Na meranie šírky pásma sa často využíva metóda *medzier testovacích paketov (probe gap method)* [20]. Na meranie kapacity hromadného prenosu je zvyčajne používaná kombinácia nástrojov *ipref* [18] a *TReno* [15].

4.3.3 Stratovosť (*packet loss*)

K stratám paketov dochádza, ak je paket zahodený buď v dôsledku chybného kontrolného súčtu (táto situácia nastáva v prípade, že je paket poškodený), alebo ak k jeho zahodeniu dôjde na niektorom zo sieťových zariadení počas jeho transportu ku

koncovej stanici, napríklad v dôsledku chybnej konfigurácie zariadenia. Stratovosť teda môžme charakterizovať ako celkový počet poškodených, alebo nedoručených paketov koncovej stanici.

4.3.4 Oneskorenie (*delay*)

Oneskorenie paketu môže byť zapríčinené rôznymi faktormi:

Oneskorenie spôsobené smerovaním

Toto oneskorenie predstavuje čas, ktorý paket potrebuje na to, aby prešiel smerovačom *(routerom)*, to znamená, že je to čas medzi príchodom posledného bitu paketu po vstupnej linke do smerovača a odoslaním prvého bitu paketu po výstupnej linke smerom k ďalšiemu zariadeniu. Tento druh oneskorenia je ďalej možné rozdeliť na [17]:

- ***Oneskorenie spôsobené spracovaním paketu (packet processing delay)*** predstavuje čas, ktorý je potrebný na spracovanie paketu t.j. určenie výstupného portu na základe cieľovej adresy a následný presun paketu cez vnútorné obvody smerovača až na tento port.
- ***Oneskorenie spôsobené čakaním vo fronte (queuing delay)*** predstavuje čas, ktorý paket strávy čakaním vo frontách smerovačov. To znamená buď čas čakania na spracovanie alebo čas čakania na odoslanie. Z pohľadu oneskorenia sú významnejšie fronty, ktoré patria k výstupným portom smerovača.
- ***Ďalšie oneskorenie (additional delay)*** predstavuje čas spôsobený inými okolnosťami, napr. nečinnosťou smerovača alebo pádom linky smerujúcej k ďalšiemu prvku siete.

Jednosmerné oneskorenie (*One Way Delay OWD*)

Predstavuje hodnotu časového posunu, ktorý uplynie od odoslania paketu zo zdrojovej stanice a prijatím toho istého paketu cieľovou stanicou [13].

Oneskorenie počas prenosu (transmission delay D_t)

Predstavuje čas, ktorý je potrebný na prenos celého objemu paketu na linku, teda od jeho prvého po posledný bit. Teda oneskorenie D_t je dané vzťahom [17]

$$D_t = s/c \tag{4.2}$$

kde s je veľkosť paketu a c je kapacita linky, po ktorej je paket prenášaný.

Oneskorenie počas šírenia (propagation delay D_p)

Predstavuje čas, za ktorý sa paket dostane z jedného konca linky na druhý. Teda dané oneskorenie D_p je možné vyjadriť vzťahom [17]

$$D_p = d/v \tag{4.3}$$

kde premenná d predstavuje dĺžku linky, po ktorej je paket prenášaný a premenná v predstavuje rýchlosť šírenia signálov danou linkou.

Oneskorenie paketu je aditívna metrika, to znamená, že jej hodnota sa môže zvyšovať postupne a každý z vyššie uvedených druhov oneskorenia sa môže vyskytovať v každom uzle siete (za uzol považujeme smerovač) na trase od zdroja až k cieľu paketu (per-hop delay). Pričom oneskorenia v jednotlivých uzloch majú aditívnu vlastnosť.

4.3.5 Kolísanie oneskorenia (*jitter*)

V kontexte počítačových sietí *jitter* popisuje kolísavosť medzičasov príchodu paketov, je to teda miera plynulosti príchodu paketov [17]. Toto časové oneskorenie vzniká počas transportu paketov počítačovou sieťou a je spôsobené čakaním na smerovanie na jednotlivých smerovačoch, prípadne nedostatočnou rpiepustnosťou liniek. Ak sú hodnoty tohto parametra malé, je možné ich na aplikačnej úrovni odstrániť a teda nevedú k veľkým problémom pri poskytovaní služieb, avšak vysoké hodnoty tohto parametra môžu viesť k problémom poskytovania QoS parametrov.

5 Návrh modulu pre agregáciu záznamov o IP tokoch

V predchádzajúcich kapitolách tejto práce bola analyzovaná problematika potreby redukcie informácií o zachytených IP tokoch v súčasných konvergovaných sieťach, ako aj jednotlivé metódy a prístupy k dosiahnutiu analyzovanej funkcionality. Ako analýza ukázala, zmeny je potrebné navrhnúť a implementovať v najnižšej, základnej vrstve nástroja SLAmeter, v nástroji MyBeem. Samotnú úpravu záznamov je nutné umiestniť práve na túto vrstvu nástroja, aby sa tak odstránila záťaž vyšších vrstiev architektúry a zredukovala sa veľkosť a množstvo zasielaných záznamov hneď na základnej úrovni spracovania nameraných hodnôt, kde je to najefektívnejšie.

5.1 Princíp modifikácie granularity záznamov

Ako už bolo v sekcii 2.1 detailnejšie popísané, úlohou meracieho procesu (obr. 2 – 2) je generovanie záznamov o zachytených tokoch, úlohou exportovacieho procesu (obr. 2 – 2) je export týchto informácií smerom k zhromažďovaciemu procesu, tak ako je to uvedené v dokumente [24]. Tento dokument však nijako bližšie nešpecifikuje dátovú štruktúru a ani metódu posielania údajov medzi týmito dvoma procesmi, ako ani to, ako nakladať s týmito záznamami, kým sú prenechané exportovaciemu procesu. Preto je vhodné umiestniť modul agregácie IP tokov práve na toto miesto, aby tak bola zabezpečená kontrola množstva záznamov o tokoch zasielaných meracím procesom a prijatých exportovacím procesom, a zároveň aby nebolo ovplyvnené odchytávanie a identifikácia IP tokov. Modul agregácie teda agreguje záznamy o tokoch v prípade, že ich počet presahuje stanovenú kritickú hodnotu resp. v prípade, ak si používateľ manuálne zvolí možnosť agregácie záznamov o tokoch nastavením príslušného parametra v konfiguračnom súbore programu MyBeem.

Návrh tohto modulu počíta s prvotným stanovením podmienok, za akých bude do-

chádzať k agregácii záznamov. Exportér si udržiava vo vyrovnávacej pamäti tokov záznam o danom toku po dobu jeho expirácie. Ak dôjde k zachyteniu koncového paketu toku, k jeho aktívnej alebo pasívnej expirácii, záznam je presunutý (resp. v prípade aktívnej expirácie je to kópia záznamu o toku) do pamäte expirovaných tokov a následne prenechaný exportovaciemu procesu a preposlaný na kolektor. V súčasnej verzii exportéra je implementovaná podpora zatrieďovania paketov do tokov na základe porovnávania kľúčových hodnôt daného toku. Niekedy je ale možné sledovať problém preplnenia vyrovnávacej pamäte tokov a to najmä pri udalostiach v sieti, kedy merací bod zachytáva veľké množstvo krátkych tokov, o ktorých si musí udržiavať záznami vo vyrovnávacej pamäti tokov po dobu ich expirácie. Tento problém sa stáva akútnym najmä v prípadoch, kedy v sieti dochádza k nejakému typu útoku (napr. DoS útok) resp. k udalosti, pri ktorej je generované veľké množstvo krátkych tokov (väčšinou sa jedná o toky, ktoré tvorí jeden paket) s rovnakou hodnotou jedného kľúčového parametra toku, ale rôznymi hodnotami ostatných parametrov. V súčasnej implementácii je každý z týchto paketov zaradený to samostatného záznamu o toku. Daný záznam je uchovaný vo vyrovnávacej pamäti po dobu jeho pasívnej expirácie, keďže sa jedná o neukončený tok s jediným paketom. Problém však nastáva pri množstve tokov, ktoré sa blíži resp. prekročí veľkosť alokovanej pamäte pre vyrovnávaciu pamäť tokov. Exportér musí byť schopný zvládať aj takýto druh prevádzky bez toho, aby dochádzalo k úplnej strate informácií o tokoch. Preto je potrebné:

- v prípade, že nedochádza k nadmernému zaťaženiu exportéra a používateľ si nezvolil inak, ponechať jeho štandardné správanie, bez akejkoľvek agregácie alebo modifikácie záznamov o tokoch
- ak však dochádza k nadmernému zaťaženiu vyrovnávacej pamäte tokov alebo si to používateľ vyžiada nastavením príslušného parametra v konfiguračnom súbore programu, je potrebné okamžite reagovať na túto udalosť spustením procesu agregácie záznamov o tokoch

5.2 Zaraďovanie paketov do tokov

V súčasnej implementácii meracieho procesu nástroja MyBeem sa zaraďovanie paketov do tokov deje vo vlákne, ktoré spracováva jednotlivé zachytené pakety. Jednoznačné zaradenie paketu do toku sa deje porovnávaním všetkých kľúčových hodnôt toku s hodnotami príslušných položiek zachyteného paketu. Jedná sa teda o päticu ukazovateľov v poradí podľa tabuľky 5 – 1.

Tabuľka 5 – 1: Porovnávané premenné pri zatrieďovaní paketov do tokov

Poradie	Premenná
1.	ipProtocol
2.	ipSrcAddress
3.	ipSrcPort
4.	ipDstAddress
5.	ipDstPort

V novej koncepcii programu je potrebné tento proces čo možno najviac urýchliť, keďže bude nutné stále pri preorganizácii vyrovnávacej pamäte tokov uzamknúť celú vyrovnávaciu pamäť, aby sa tak zamedzilo prístupu ostatných vlákien. Toto uzamknutie ale spôsobí, že vo vyrovnávacej pamäti pre nezaradené pakety sa budú tieto pakety hromadiť, pokiaľ neprebehne celý proces preusporiadania vyrovnávacej pamäte a pamäť nebude znova sprístupnená pre modifikáciu aj ostatným procesom. Jednou z efektívnejších možností by bolo zatrieďovanie paketov podľa jednoznačného identifikátora paketu voči toku *packet_flow_identificator*, ktorý by paket priradil do príslušného toku na základe výpočtu hašovanej hodnoty z pätice kľúčových hodnôt toku za pomoci hašovacej funkcie, resp. v prípade, že by daný záznam o toku vo vyrovnávacej pamäti tokov neexistoval, bol by vytvorený a každým ďalším prichádzajúcim paketom aktualizovaný. Je teda samozrejmé, že pre výpočet identifikátora toku a identifikátora príslušnosti paketu voči toku musí byť použitá rovnaká hašo-

vacia funkcia, aby bolo možné túto príslušnosť jednoznačne identifikovať. Pri tomto spôsobe identifikácie príslušnosti paketu voči toku bude teda nutné vykonať len jednu operáciu porovnania, a to porovnanie identifikátora toku s identifikátorom paketu *packet_flow_identificator*.

Menšia komplikácia však nastáva, pri zapnutej identifikácii obojsmerných tokov,

Tabuľka 5 – 2: Poradie premenných zreťazených na vstupe do hašovacej funkcie pri tvorbe identifikátora paketu voči toku

Poradie	Premenná (*forward direction*)	Premenná (*backward direction*)
1.	ipProtocol	ipProtocol
2.	ipSrcAddress	ipDstAddress
3.	ipDstAddress	ipSrcAddress
4.	ipSrcPort	ipDstPort
5.	ipDstPort	ipSrcPort

čo znamená že toky smerujúce od zdroja k cieľu a od cieľovej stanice k zdrojovej sú identifikované ako jeden tok. Súčasná implementácia ukazovateľa *flowId* by tieto toky identifikovala ako dva rozdielne, čo platí aj pre pakety, ktoré by do týchto záznamov prichádzali. Je to zapríčinené tým, že do hašovacej funkcie síce vstupujú rovnaké hodnoty, avšak v rozdielnom poradí, čo spôsobuje, že je hašovaný rozdielny reťazec (podľa tabuľky 5 – 2).

Tabuľka 5 – 3 zobrazuje porovnanie hodnôt tvoriacich reťazec pre hašovanú hodnotu pri pakete smerujúcom od zdroja k cieľu a toku, ktorý je identifikovaný ako smerujúci od cieľa k zdroju, ako aj samotné reťazce vstupujúce do hašovacej funkcie pre tvorbu identifikátora *flowId*.

Teda pre efektívne fungovanie navrhovaného mechanizmu zatrieďovania paketov do tokov a pri zapnutej identifikácii obojsmerných tokov *biflows*, bude potrebné vypočítať ešte jeden haš *bwd_packet_flow_identificator* použitím tej istej hašovacej funkcie, aby bolo možné identifikovať, či sa nejedná o paket spätného toku. Tento výpočet

Tabuľka 5 – 3: Tvorba identifikátora *flowId* pre tok a spätný tok

Kľúčové hodnoty	Identifikátory toku (smer zdroj→cieľ)	Identifikátory paketu (smer cieľ→zdroj)
protocolIdentifier	ICMP	ICMP
sourceIPv4Address	10.0.2.15	173.194.35.148
destinationIPv4Address	173.194.35.148	10.0.2.15
sourceTransportPort	5190	2323
destinationTransportPort	2323	5190
Reťazec vstupujúci do hašovacej funkcie	ICMP10.0.2.15173. 194.35.14851902323	ICMP173.194.35.14810. 0.2.1523235190
Výsledné hodnoty flowId	2990645315802650056	10901413210712831870

prebehne iba v prípade, že sa nenájde zhoda pri prvom porovnaní identifikátora *flowId* a identifikátora *packet_flow_identificator* a je zapnutá podpora identifikácie obojsmerných tokov *biflows*. V tabuľke 5 – 3 je zdrojová a cieľová IP adresa zapísaná v klasickom tvare, rozdelená do štyroch 8-bitových čísel, pre lepšiu čitateľnosť. V programe však budú do hašovacej funkcie vstupovať IP adresy ako 32-bitové čísla. Navrhovaná metóda si stále vyžaduje omnoho menej porovnaní, ako súčasná implementovaná metóda a zároveň je možné hašovaný identifikátor paketu použiť pre zaradenie toku, ktorý tento paket vytvoril, do vyrovnávacej pamäte a teda nieje nutné ho počítať znova. Táto metóda si vyžaduje minimálne 1 a maximálne 2 porovnania pre zaradenie paketu, kdežto súčasná metóda vyžaduje minimálne 5 a maximálne 10 porovnaní. Pri vypnutej identifikácii obojsmerných tokov *biflows* nemusí byť vypočítaný aj identifikátor pre paket spätného toku, pretože toky smerujúce od zdroja k cieľu a od cieľa k zdroju sú identifikované ako dva samostané toky.

5.3 Stanovenie kritickej hodnoty

Ešte pred samotným popisom procesu agregácie v programe MyBeem je nutné zadefinovať kritickú hodnotu stavu vyrovnávacej pamäte, pri ktorej dôjde k spusteniu agregácie tokov, ak si používateľ nevyžiada spustenie agregačného procesu už pri štarte programu.

Ak je v sieti nadmerné množstvo tokov, je možné, že ich exportér nedokáže spracovávať ani napriek dostatočne nízkej hodnote parametra aktívnej alebo pasívnej expirácie. Jedným z možných riešení by bolo zníženie časového intervalu pre pasívnu alebo aktívnu expiráciu tokov, čo by ale mohlo spôsobiť zbytočné zvýšenie frekvencie exportovania záznamov o tokoch zhromažďovacej aplikácii a efektívne by to nemuselo priniesť požadovaný výsledok. Tým ale vzniká nebezpečenstvo zlyhania vyšších vrstiev architektúry, pretože môže dôjsť k preťaženiu kolektora, alebo databázy resp. môže dôjsť k vyčerpaniu šírky pásma linky smerujúcej od exportéra k vyšším vrstvám, ak by sa nejednalo o modernú vysokorýchlostnú linku. Nehovoriac o tom, že týmto správaním sa aplikácie iba problém posúvame vyšším vrstvám architektúry, teda ho žiadnym spôsobom efektívne neriešime. Efektívnym spôsobom riešenia daného problému by bolo zahájenie procesu agregácie tokov. Preto je potrebné stanoviť kritickú hodnotu *threshold T*, pri ktorej sa zaháji proces agregácie. Teda platí, že k agregácii dôjde ak:

$$L \geq T \tag{5.1}$$

pričom platí, že *L (load)* vyjadruje aktuálnu hodnotu vyťaženia vyrovnávacej pamäte a *T (threshold)* vyjadruje hraničnú hodnotu vyťaženia, pri ktorej dôjde k spusteniu agregačného mechanizmu. Obe veličiny nadobúdajú hodnoty z intervalu *<0,1>*.

5.4 Spúšťanie procesu agregácie

Pri agregácii tokov je potrebné určiť nielen to, kedy k agregácii začne dochádzať (stanovenie *kritickej hodnoty*), ale tiež to, ktoré toky budú tomuto procesu pod-

liehať, a ako často sa bude vykonávať proces preogranizácie vyrovnávacej pamäte tokov. Preto zavedieme veličinu, ktorú budeme nazývať spúšťač agregácie *aggregation trigger* A_t.

Aggregation trigger A_t je časový interval v milisekundách, po uplynutí ktorého bude opakovane dochádzať k spúšťaniu procesu agregácie záznamov o tokoch vo vyrovnávacej pamäti tokov.

Proces agregácie nemôže bežať nad vyrovnávacou pamäťou tokov neustále, pretože by nebolo možné efektívne selektovať dátovo rozsiahle toky, ktoré nebudú modifikované. Problémom je, že za tak krátky časový okamih merací proces nemusí zachytiť relevantné množstvo informácií o danom toku, na základe ktorých by bolo možné rozhodnúť či sa jedná o rozsiahly tok.
Okrem stanovenia frekvencie s akou bude dochádzať k spúšťaniu agregačného procesu je nutné stanoviť, ktoré toky budú tomuto procesu podliehať. V nasledujúcej kapitole je vypracovaný návrh selekcie týchto tokov.

5.5 Identifikácia rozsiahlych tokov

V procese agregácie bude potrebné určiť, ktoré toky budú považované za rozsiahle a teda nedôjde k ich agregácii. Zároveň sa týmto spôsobom identifikujú ostatné toky, ktoré budú podliehať procesu agregácie. Preto teraz navrhneme spôsob identifikácie rozsiahlych tokov, ktorý bude použitý pri implementácii v tejto diplomovej práci.

Rozsiahlym tokom v procese agregácie bude ten záznam, ktorého hodnota počtu oktetov *octet total count* prekročí stanovenú hranicu za časový interval rovný hodnote premennej *aggregation trigger* A_t.

Z predošlej definície vyplíva, že naopak agregácii budú podliehať toky, ktoré nebudú identifikované ako rozsiahle, teda pri ktorých:

$$octetTotalCount < A_c \tag{5.2}$$

Veličina *octet total count* je informačný element, ktorý v danom čase určuje, koľko oktetov daného toku bolo zachytených monitorovacím procesom.

Aggregation condition A_c predstavuje hraničnú hodnotu počtu oktetov toku F_x, ktorú ak veličina *octet total count* neprekročí za časový interval stanovený ako *aggregation trigger* A_t, tok bude považovaný za malý, a teda dôjde k jeho agregácii.

Pre určenie počiatočných hodnôt veličín A_t a A_c, ktoré budú primárne zadané v konfiguračnom súbore programu po jeho stiahnutí, je potrebné vykonať pokusné merania. Zároveň je však potrebné umožniť používateľovi, aby si mohol nastaviť túto hodnotu podľa vlastných preferencií.
Týmto spôsobom vieme efektívne stanoviť hodnoty charakteristické pre toky, ktoré chceme podrobiť agregačnému procesu a zároveň tak identifikovať rozsiahle toky. V konečnom dôsledku musí byť hodnota veličiny *aggregation trigger* A_t menšia, ako veľkosť časového intervalu pre aktívnu expiráciu toku (pričom veľkosť časového intervalu pre aktívnu expiráciu toku musí byť menšia ako veľkosť intervalu pre jeho pasívnu expiráciu).
Niektoré štúdie sa zaoberajú myšlienkou použitia vzorkovania paketov ako účinnej techniky na určovanie rozsiahlych tokov. Z tohto dôvodu boli vykonané pokusné merania, pri ktorých bol kladený dôraz na určovanie roziahlych tokov vzorkovacou technikou. Výsledok je rozpracovaný v nasledujúcej sekcii.

Zhodnotenie použiteľnosti vzorkovania paketov pre potreby identifikácie rozsiahlych tokov

V sekcii 4.1 boli popísané rôzne techniky vzorkovania, ktoré sú implementované v programe MyBeem. Pre účely agregácie informácií o IP tokoch je vhodné preskúmať spoľahlivosť jednotlivých vzorkovacích techník pri určovaní rozsiahlych tokov v porovnaní s technikou navrhovanou v tejto diplomovej práci. Implementácia vzorko-

vacej techniky v programe MyBeem musela byť mierne upravená tak, aby vyhovovala účelu, ktorý bolo potrebné odtestovať. Po odchytení paketu meracím procesom sa paket podrobil vzorkovaciemu algoritmu. V prípade, že bola splnená podmienka, paket bol označený a následne pri priraďovaní do toku bola táto značka nahratá do premennej v príslušnom zázname o toku, ktorá bola vypísaná na konzolový výstup programu. V prípade nevyhovenia podmienky nebol paket zahodený, ako to býva v klasickej implementácii vzorkovacích metód, ale bol taktiež zaradený do toku.
Vo výstupoch jednotlivých metód je potrebné si všímať ukazovatele súčtu oktetov z označených paketov toku *SAMPLED Octet total count* a porovnať ich s ukazovateľom celkového počtu oktetov toku *Octet total count*. Totižto ak by bola táto metóda použitá v pocese agregácie navrhovanom v tejto práci, ukazovateľ súčtu oktetov z označených paketov *SAMPLED Octet total count* by bol porovnávaný s podmienkou agregácie toku o počte oktetov *aggregation condition* A_c. Z príslušných meraní bolo vybraných niekoľko záznamov o tokoch.

Systematic count based sampling

Pri prevedení pokusu s týmto typom vzorkovania paketov bola hodnota *samplingPacketInterval*, teda počet za sebou nasledujúcich označených paketov, stanovená na 50 paketov v rade a hodnota *samplingPacketSpace*, teda počet za sebou nasledujúcich neoznačených paketov, na 15 paketov v rade. Z obrázka 5 – 1 je zrejmé, že ak by sme stanovili hranicu agregácie na 100000 oktetov v toku, čo reprezentuje 0,1% kapacity 100 Mbps linky, tento proces by s veľkou pravdepodobnosťou chybne neidentifikoval veľký tok. Tento problém sa vyskytoval častejšie na skúmanej vzorke.

Systematic time based sampling

Pri monitorovaní prevádzky s označovaním paketou touto technikou bola hodnota ukazovateľa *samplingTimeInterval*, teda časový interval, v ktorom budú pakety značkované, stanovená na 3 sekundy a hodnota ukazovateľa *samplingTimeSpace*, teda časový interval, v ktorom nebude dochádzať k značkovaniu paketov na 0,5 sekundy. Z

```
29.4.2014  21:47:6  root  Beem  INFORMATION  Flow_ID value: 1357418663453016418
29.4.2014  21:47:6  root  Beem  INFORMATION  Source MAC -> 0:25:b4:da:99:c0
29.4.2014  21:47:6  root  Beem  INFORMATION  Expiration reason value -> 2
29.4.2014  21:47:6  root  Beem  INFORMATION  Octet total count -> 128752
29.4.2014  21:47:6  root  Beem  INFORMATION  SAMPLED Octet total count -> 99964
29.4.2014  21:47:6  root  Beem  INFORMATION  Type of used protocol -> [TCP]
29.4.2014  21:47:6  root  Beem  INFORMATION  Source IP address -> 80.94.52.21
29.4.2014  21:47:6  root  Beem  INFORMATION  Destination IP address -> 147.232.48.50
29.4.2014  21:47:6  root  Beem  INFORMATION  Packet total count -> 46
29.4.2014  21:47:6  root  Beem  INFORMATION  Speed -> 22 kBps
29.4.2014  21:47:6  root  Beem  INFORMATION  Amount of DATA sent -> 128752:0
```

Obrázok 5 – 1: Systematic count based sampling

obrázka 5 – 2 je zrejmé, že daná technika taktiež nedokázala spoľahlivo identifikovať rozsiahle toky, keď označila iba 466 oktetov z celkovej veľkosti 102857 oktetov toku. Ak by sme ju mali porovnať s technikou identifikácie rozsiahlych tokov navrhovanou v tejto práci, ktorá určuje rozsiahly tok na základe pevného stanovenia agregačnej podmienky počtu oktetov *aggregation condition* A_c a následného pravidelného prehľadania vyrovnávacej pamäte a agregácii tokov po uplynutí stanoveného časového intervalu, tak metóda vzorkovania *Systematic time based sampling* nespĺňa stanovené kritériá.

```
2014  22:1:37  root  Beem  INFORMATION  Flow_ID value: 16196328645641252472
2014  22:1:37  root  Beem  INFORMATION  Source MAC -> 0:25:b4:da:99:c0
2014  22:1:37  root  Beem  INFORMATION  Expiration reason value -> 3
2014  22:1:37  root  Beem  INFORMATION  Octet total count -> 102857
2014  22:1:37  root  Beem  INFORMATION  SAMPLED Octet total count -> 466
2014  22:1:37  root  Beem  INFORMATION  Type of used protocol -> [TCP]
2014  22:1:37  root  Beem  INFORMATION  Source IP address -> 85.248.228.67
2014  22:1:37  root  Beem  INFORMATION  Destination IP address -> 147.232.48.50
2014  22:1:37  root  Beem  INFORMATION  Packet total count -> 31
2014  22:1:37  root  Beem  INFORMATION  Speed -> 57 kBps
2014  22:1:37  root  Beem  INFORMATION  Amount of DATA sent -> 102857:0
```

Obrázok 5 – 2: Systematic time based sampling

Random n of N sampling

Pri tomto type vzorkovania bola hodnota vyberaného počtu paketov n stanovená na 200 paketov a veľkosť rodičovskej množiny N na 260 paketov. Z obrázka 5 – 3 je zrejmé, že daná metóda neidentifikovala správne rozsiahle toky. Ak by sme agre-

```
29.4.2014  22:5:46  root  Beem  INFORMATION  Flow_ID value: 1439311275577276154
29.4.2014  22:5:46  root  Beem  INFORMATION  Source MAC -> 0:25:b4:da:99:c0
29.4.2014  22:5:46  root  Beem  INFORMATION  Expiration reason value -> 3
29.4.2014  22:5:46  root  Beem  INFORMATION  Octet total count -> 103901
29.4.2014  22:5:46  root  Beem  INFORMATION  SAMPLED Octet total count -> 22448
29.4.2014  22:5:46  root  Beem  INFORMATION  Type of used protocol -> [TCP]
29.4.2014  22:5:46  root  Beem  INFORMATION  Source IP address -> 81.2.197.144
29.4.2014  22:5:46  root  Beem  INFORMATION  Destination IP address -> 147.232.48.50
29.4.2014  22:5:46  root  Beem  INFORMATION  Packet total count -> 70
29.4.2014  22:5:46  root  Beem  INFORMATION  Speed -> 29 kBps
29.4.2014  22:5:46  root  Beem  INFORMATION  Amount of DATA sent -> 103901:0
```

Obrázok 5 – 3: Random n of N sampling

gačnú podmienku *aggregation condition* A_c stanovili na 100000 oktetov v toku (0,1% kapacity 100Mbps linky), proces by v prípade daného rozsiahleho toku identifikoval iba 22448 oktetov a tým by tento tok chybne určil ako malý, takže tento tok by bol v ďalšom procese chybne agregovaný. Tento tok má však celkový počet oktetov v danom exporte rovný 103901, čiže musí byť identifikovaný ako rozsiahly.

Uniform probability sampling

Pri tejto metóde bola pravdepodobnosť výberu z množiny odchytených paketov stanovená na hodnotu 85%. V skúmanej vzorke sa zistilo, že podobne ako predošlá metóda, tak aj táto v niektorých prípadoch neoznačila veľké toky (obr.5 – 4). To znamená, že pri špecifikovanom nastavení agregačnej podmienky na 100000 oktetov v toku by daná metóda taktiež nebola spoľahlivá.

```
29.4.2014  22:44:46  root  Beem  INFORMATION  Flow_ID value: 16179159175675939917
29.4.2014  22:44:46  root  Beem  INFORMATION  Source MAC -> 0:25:b4:da:99:c0
29.4.2014  22:44:46  root  Beem  INFORMATION  Expiration reason value -> 2
29.4.2014  22:44:46  root  Beem  INFORMATION  Octet total count -> 110760
29.4.2014  22:44:46  root  Beem  INFORMATION  SAMPLED Octet total count -> 68404
29.4.2014  22:44:46  root  Beem  INFORMATION  Type of used protocol -> [TCP]
29.4.2014  22:44:46  root  Beem  INFORMATION  Source IP address -> 80.94.52.28
29.4.2014  22:44:46  root  Beem  INFORMATION  Destination IP address -> 147.232.48.50
29.4.2014  22:44:46  root  Beem  INFORMATION  Packet total count -> 21
29.4.2014  22:44:46  root  Beem  INFORMATION  Speed -> 24 kBps
29.4.2014  22:44:46  root  Beem  INFORMATION  Amount of DATA sent -> 110760:0
```

Obrázok 5 – 4: Uniform probability sampling

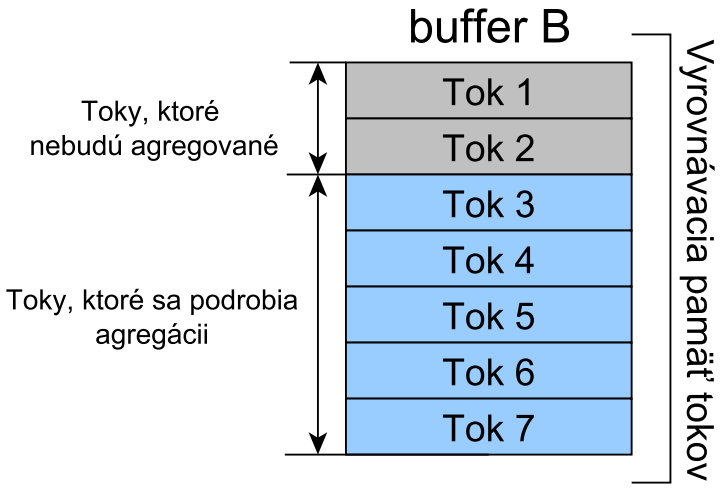
Sumarizácia výsledkov testovania vzorkovacích algoritmov

Týmto pokusom bola preukázaná nevhodnosť použitia vzorkovania pre identifikáciu rozsiahlych tokov pre účely tejto diplomovej práce, pretože ani jedna zo skúmaných techník nedokázala efektívne a spoľahlivo identifikovať rozsiahle toky v každom z prípadov. Samozrejme cieľom tohto pokusu nieje spochybniť účinnosť vzorkovacích algoritmov. Výsledok naznačuje len nevhodnosť použitia vzorkovania pre účely agregácie, pretože môže dochádzať k zásadným stratám údajov pri chybnej identifikácii tokov. Tým sa potvrdili predpoklady niektorých štúdií, že použitím klasických vzorkovacích metód pre účely identifikácie rozsiahlych tokov dochádza k značnej chybovosti.
Riešením by bolo nastavenie daných algoritmov tak, aby označovali veľké percento skúmanej vzorky, avšak vtedy by bola podstata vzorkovania, ako selektívnej metódy, do veľkej miery potlačená, keďže by sa jednalo o označovanie takmer všetkých odchytených paketov a nejednalo by sa o skúmanú "vzorku".

5.6 Postupnosť krokov pri agregácii

Na základe analýzy vypracovanej v predošlých kapitolách tejto diplomovej práce navrhneme jednotlivé kroky agregačného procesu. V konfiguračnom súbore programu je potrebné určiť priority jednotlivých kľúčových hodnôt toku, tak ako je to znázornené v tabuľke 3 – 1, podľa ktorých sa bude pristupovať k postupnej redukcii týchto hodnôt v procese agregácie. Hodnoty priorít sa načítajú z konfiguračného súboru do programu pri jeho inicializácii. Zároveň je potrebné určiť, ktoré toky budú agregované, a ktoré budú exportované bez zásahu a teda ponechané v pôvodnej nezmenenej podobe (v bufferi B)(obr.5 – 5). Pre tento účel nám poslúži práve veličina *aggregation condition* A_c. Porovnávanie aktuálneho stavu s touto hodnotou sa bude vykonávať v každom z bufferov spomínaných nižšie v tejto sekcii pre každý tok, či už bude agregovaný alebo nie.

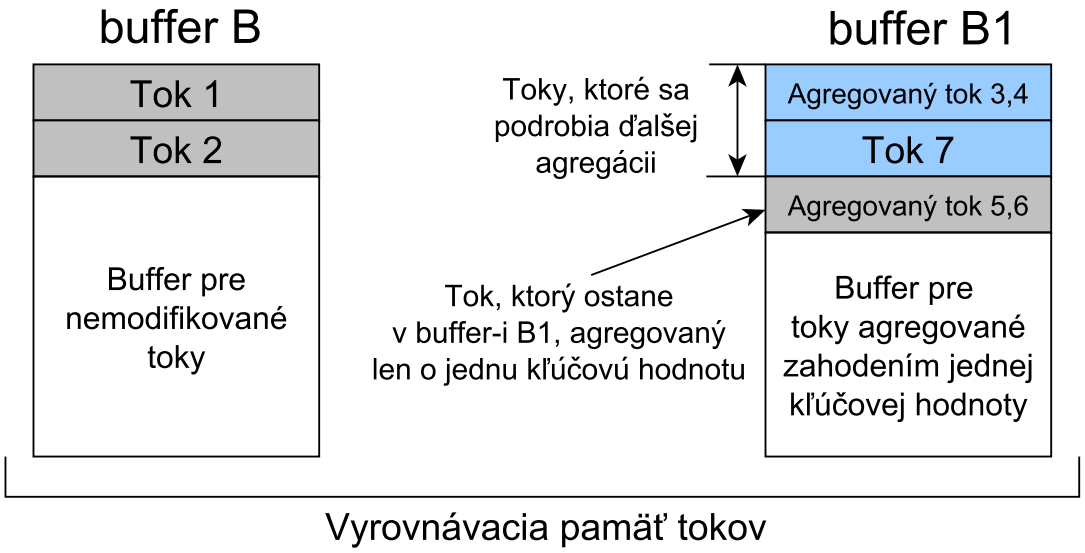
Poznamka: Pomenovanie "*buffer*" v tejto aj ďalších sekciách diplomovej práce označuje spájaný hašovaný zoznam, do ktorého budú zaraďované referencie na štruktúry obsahujúce údaje o neagregovaných alebo agregovaných tokoch.



Obrázok 5 – 5: Vyrovnávacia pamäť tokov (buffer B)

Pre toky, ktoré sa vyčlenia algoritmom na agregáciu je potrebné vytvoriť buffer B_1, do ktorého sa budú ukladať, a z ktorého budú presunuté po uplynutí aktívnej alebo pasívnej expirácie do pamäte expirovaných tokov a exportované zhromažďovaciemu procesu. Zároveň sú originálne záznamy po ich presune do nového buffera B_1 redukované o jednu kľúčovú hodnotu a pri ich prípadnom zlúčení s už existujúcim záznamom v tomto bufferi zahodené a ponechaná je len agregovaná hodnota záznamov o tokoch v novom bufferi B_1(obr.5 – 6). Týmto síce záznamy strácajú svoju úplnú detailnosť, avšak agregácia skupín malých tokov vo väčšine prípadov analýzy meranej prevádzky nepredstavuje veľký problém, ktorý by výrazným spôsobom ohrozil kvalitatívnu stránku analytických záverov.

Následne je nutné po uplynutí času, definovaného ako *aggregation trigger* A_t, opäť vykonať kontrolu vyťaženia pamäte tak, ako je to popísané v sekcii 5.4, avšak tento-

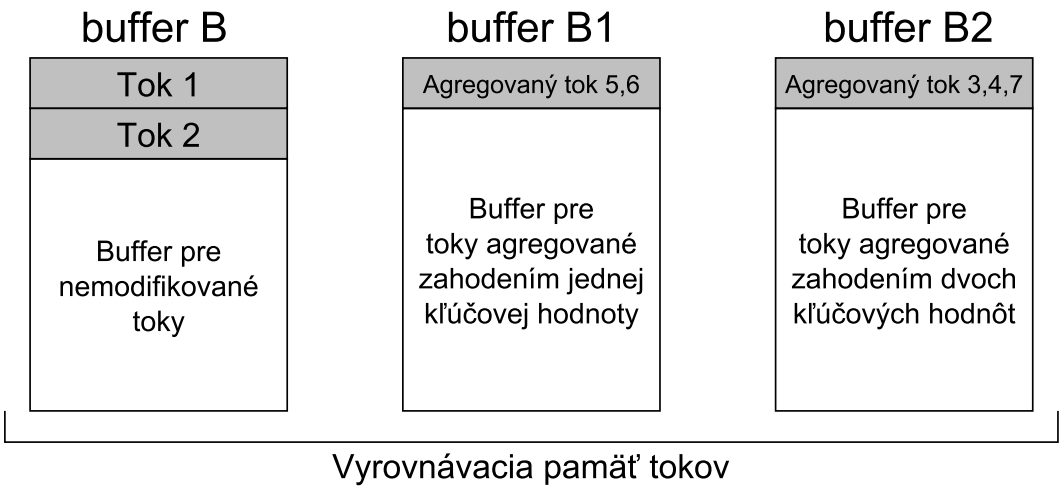


Obrázok 5 – 6: Vyrovnávacia pamäť tokov (buffer B a B_1)

raz už v oboch bufferoch. Ak sa po kontrole ukáže, že buffer B_1 aj naďalej obsahuje toky, ktoré nespĺňajú podmienku počtu oktetov *aggregation condition* A_c, je nutné opakovane pristúpiť k podobnému postupu. Pre buffer B_1 je teda nutné vytvoriť ďalší buffer B_2. To znamená vyčleniť z buffera B_1 pre používateľa dôležitejšie toky, ktorých agregačný stupeň spočíva v redukovaní jednej kľúčovej hodnoty toku (obr.5 – 6). Ostatné toky je potrebné umiestniť do buffera B_2 a vykonať agregáciu ďalšieho stupňa, t.j. okresať kľúčové hodnoty toku o ďalšiu hodnotu.

Tento spôsob je potrebné opakovať, teda pre každú ďalšiu kombináciu kľúčových hodnôt vytvoriť nový buffer, až pokým sa nedosiahne najvyšší stupeň agregácie, kedy ostáva iba jedna určujúca kľúčová hodnota toku.

Takýmto spôsobom je možné veľmi ľahko zabrániť preťaženiu exportéra v dôsledku preplnenia vyrovnávacej pamäte tokov, pretože pre všetky malé toky vyskytujúce sa v niektorom z bufferov a zaberajúce n položiek vyrovnávacej pamäte bude vytvorená *jedna* položka v nasledujúcom bufferi, do ktorej budú všetky tieto malé toky agregované, pričom pôvodné záznamy budú zahodené a tým sa vytvorí značný počet voľných položiek pre uloženie nových záznamov o tokoch.



Obrázok 5 – 7: Vyrovnávacia pamäť tokov (buffer B, B_1 a B_2)

Typickým príkladom generovania obrovského množstva malých tokov, ktoré majú spoločnú cieľovú IP adresu, ale rozdielne zdrojové adresy je DoS útok, ktorý môže byť smerovaný na zariadenie nachádzajúce sa v sieti za meracím bodom. Práve táto situácia by starej koncepcii nástroja spôsobila nemalé problémy. V novej koncepcii bude možné agregáciou podľa cieľovej IP adresy toto enormné množstvo tokov zachytených meracím bodom agregovať do jedného a tým zabrániť zrúteniu meracieho procesu. Zároveň je možné tento útok exportérom veľmi ľahko detekovať a upozorniť tak používateľa na možné riziko alebo vytvoriť záznam na logovacom zariadení o tejto udalosti, kedy k útoku došlo.

5.7 Umiestnenie vytvorených záznamov o tokoch

Ako už bolo v predošlých kapitolách spomenuté, agregácii bude nutné podrobiť len isté skupiny tokov, ktorých hodnota parametra *octet total count* nebude presahovať hodnotu *aggregation condition* A_c. Práve kvôli problému identifikácie rozsiahlych tokov nemôžu byť pakety priraďované po ich odchytení aj do agregovaných tokov ostatných bufferov, ale stále musia byť vytvárané toky v základnom bufferi B (obr.

5 – 5).

Totižto, ak je rozsiahly tok odchytený tesne pred uplynutím časového intervalu *aggregation trigger* A_t a je v ňom do začiatku procesu agregácie odchytených menej oktetov, ako určuje agregačná podmienka minimálneho počtu oktetov *aggregation condition* A_c, bude tento tok chybne agregovaný. Takto môže byť jeho záznam zlúčený s inými tokmi v bufferi, kde sú toky agregované o jednu alebo viac kľúčových hodnôt. Vytvorený je teda jeden agregovaný tok, do ktorého by spadal každý ďalší odchytený paket tohto rozsiahleho toku, až pokiaľ by nedošlo k jeho pasívnej expirácii. Expiráciu toku by zároveň predlžovalo aj to, že do daného agregovaného toku by mohli byť zaraďované pakety z viacerých predtým neagregovaných tokov, a teda tok by neustále expiroval iba aktívne.

Ďalší problém by mohol nastať v prípade, ak sa v poslednom bufferi B_4 vytvorí jeden agregovaný tok, ktorý by určovala iba jedna kľúčová hodnota. V prípade, že by touto poslednou určujúcou hodnotou bol typ prenosového protokolu toku, spadali by do tohto agregovaného toku všetky pakety, ktoré by nenašli im prislúchajúce toky v niektorom z predošlých bufferov. Takto by nielenže mohlo dochádzať k chybnému zaradeniu paketov rozsiahlych tokov, ale taktiež k chybnému zaradeniu paketov menej rozsiahlych tokov, ktoré by na prvom stupni agregácie po ich zlúčení s ostatnými menej rozsiahlymi tokmi mohli prekročiť hraničný počet oktetov stanovený v *aggregation condition* A_c. Tak by sa tento tok stal rozsiahlym, aj keď už agregovaným tokom.

Preto bude potrebné nastaviť spôsob priraďovania paketov do tokov tak, aby toto priraďovanie prebiehalo iba v základnom bufferi B (obr. 5 – 5). Takto aj keď dôjde k skôr popísanej chybnej identifikácii rozsiahleho toku, kedy bude tok odchytený tesne pred uplynutím časového intervalu pre začiatok agregácie *aggregation trigger* A_t, analyzujúca aplikácia príde najviac o počet oktetov daného rozsiahleho toku, ktorý je rovný:

$$O_x = A_c - P_o \tag{5.3}$$

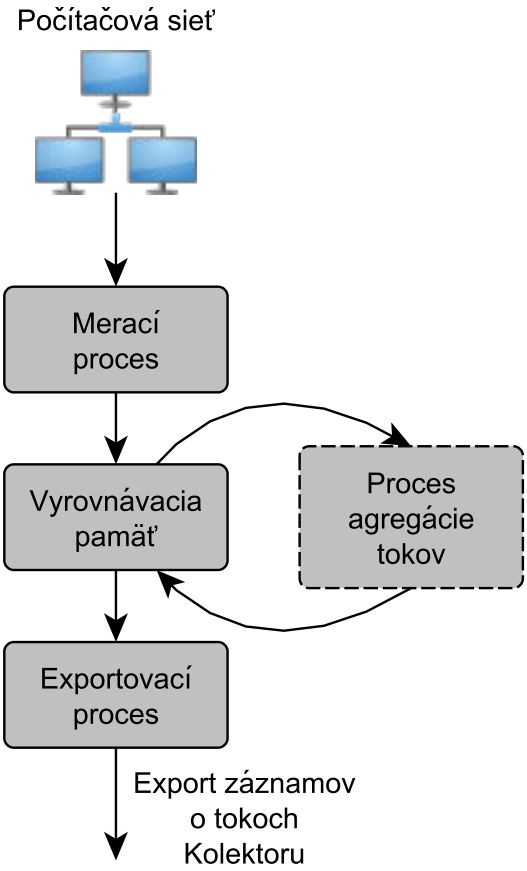
pričom ukazovateľ O_x reprezentuje maximálny počet oktetov, ktoré môžu byť chybne zaradené a uakzovateľ P_o reprezentuje počet oktetov jedného paketu daného chybne zaradeného toku. Je však potrebné pripomenúť, že údaje o daných oktetoch sa úplne nestrácajú, pretože dané oktety niesú zahodené, sú len chybne zaradené v agregovanom toku, pričom by agregované byť nemali. Spôsobom navrhovaným v tejto práci sa síce stopercentne nevyhneme chybe v prvom agregačnom cykle, avšak v ďalšom agregačnom cykle už tento daný tok bude pri rovnakej intenzite prenášaných údajov s určitosťou identifikovaný ako rozsiahly. Skôr analyzovaná technika *vzorkovania paketov* by však ani v druhom agregačnom cykle nemusela spoľahlivo identifikovať tento tok ako rozsiahly. Počet chybne zaradených paketov zároveň závisí aj od vzájomne súvisiaceho nastavenia hodnôt *aggregation condition* A_c a *aggregation trigger* A_t. Totižto ak bude hodnota parametra *aggregation trigger* A_t príliš malá a hodnota *aggregation condition* A_c príliš veľká, je možné, že žiadny z tokov nebude identifikovaný ako rozsiahly, pretože za daný časový interval analyzujúca aplikácia nemusí stihnúť zachytiť dostatočný počet oktetov daného toku pre nesplnenie agregačnej podmienky, a teda bude agregovaný takmer každý zo zachytených tokov. Ak bude hodnota *aggregation condition* A_c príliš malá, je možné, že nebude dochádzať k takmer žiadnej, alebo len k nepatrnej agregácii záznamov o tokoch.

6 Implementácia modulu agregácie a jeho procesov

Modul agregácie a celá jeho funkcionalita boli implementované v najnižšej vrstve SLAmetra, v nástroji MyBeem, ktorého charakteristika bola popísaná v sekcii 2.1. Implementácia tohto modulu a jeho jednotlivých obslužných funkcií sa primárne nachádza na rozhraní medzi meracím a exportovacím procesom (celý proces prebieha s opakovaním nad vyrovnávacou pamäťou po uplynutí časového intervalu - *aggregation trigger* A_t, obr. 6 – 1), pričom niektoré úpravy si vyžiadali taktiež zásahy aj do samotného meracieho ako aj exportovacieho procesu. V danom prípade sa jedná o rozšírenie funkcionality existujúceho nástroja, takže koncepcia a štruktúra nástroja MyBeem bola zachovaná.

Keďže tento nástroj je implementovaný v programovacom jazyku C, celý modul agregácie a taktiež aj jeho funkcionality sú riešené v tomto programovacom jazyku. Pre umiestnenie hlavnej časti zdrojového kódu modulu agregácie boli vytvorené nové zdrojové súbory *aggregation.c* a jemu prislúchajúci hlavičkový súbor *aggregation.h*, aby tak bolo zabezpečené lepšie logické členenie programu na jeho jednotlivé moduly. Implementácia si samozrejme vyžadovala aj modifikáciu už vytvorených zdrojových súborov a ich doplnenie.

V programe MyBeem funguje súčasne niekoľko vlákien, ktoré zabezpečujú jeho korektnú funkcionalitu. Je to vlákno pre časovú expiráciu tokov, vlákno pre spracovávanie paketov, vlákno pre odchytávanie paketov, vlákno pre zabezpečenie exportu tokov, a vlákno zabezpečujúce synchronizáciu so synchronizačným serverom v prípade merania charakteristík jednosmerného oneskorenia OWD. K týmto už vytvoreným vláknam bolo pridané aj vlákno pre agregáciu tokov. Vytvorenie samostatného vlákna pre proces agregácie zabezpečilo jeho spoľahlivú funkcionalitu, aby bezdôvodne nedochádzalo k prepusteniu záznamov o tokoch, ktoré je nutné agregovať. Zároveň bolo potrebné nastaviť synchronizáciu tohto vlákna s ostatnými vláknami



Obrázok 6 – 1: Zaradenie procesu agregácie medzi ostatné procesy programu MyBeem

v programe. Samotné vlákno sa pri zapnutej agregácii spúšťa pri začatí celého procesu odchytávania paketov a ukončuje sa spolu s ostatnými vláknami pri ukončení odchytávania. Ak nieje zapnutá žiadna z foriem agregácie, vlákno nie je vytvorené.

6.1 Rozšírenie konfiguračného súboru

Konfiguračný súbor programu MyBeem *config.xml* obsahuje množstvo nastavení, ktoré sa do programu nahrávajú pri jeho inicializácii. Je to takpovediac komuni-

kačný článok medzi používateľom a programom, kde môže používateľ ľahko modifikovať jeho správanie bez nutnosti zásahu do zdrojového kódu programu.

Do tohto konfiguračného súboru bola pridaná aj sekcia pre modul agregácie s náz-

```
<aggregation>
    <aggregationTrigger>3000</aggregationTrigger>
    <octetTotalCountForAggregation>100000</octetTotalCountForAggregation>
    <doAggregation>false</doAggregation>
    <automaticAggregation>false</automaticAggregation>
    <first>11</first>
    <second>7</second>
    <third>1228</third>
    <fourth>827</fourth>
</aggregation>
```

Obrázok 6 – 2: Doplnenie konfiguračného súboru o sekciu *aggregation*

vom *aggregation* (obr. 6 – 2). Táto sekcia obsahuje ako prvú položku premennú *aggregationTrigger*, do ktorej používateľ môže zadať požadovaný časový interval, po ktorom bude opakovane nastávať proces agregácie záznamov o tokoch, jedná sa o navrhovanú premennú *aggregation trigger* A_t. Ďalšou položkou je premenná *octetTotalCountForAggregation*. ktorá reprezentuje hraničný počet oktetov v toku, ktorý už používateľ považuje za rozsiahly, a teda nechce aby bol agregovaný. Jedná sa o navrhovanú premennú *aggregation condition* A_c. Ďalšou položkou tejto sekcie je premenná *doAggregation*, ktorá ak je nastavená na hodnotu *true*, tak zabezpečí spustenie celého procesu agregácie, ak je nastavená na hodnotu *false*, agregácia nebude spustená. Ďalšou z položiek je premenná *automaticAggregation*, ktorá ak je nastavená na hodnotu *true* a zároveň je klasická agregácia tokov vypnutá, zabezpečí spustenie agregačného procesu v prípade, že dôjde k naplneniu vyrovnávacej pamäte na viac ako 75% jej alokovaného priestoru. Ak je aj táto premenná nastavená na hodnotu *false*, samozrejme spolu s premennou *doAggregation*, nedôjde k spusteniu agregačného procesu. Za touto položkou nasledujú položky kľúčových hodnôt toku, ktoré budú postupne v procese agregácie redukované, v poradí od piatej položky po prvú. Jednotlivé kľúčové hodnoty sú reprezentované číslami informačných elementov a ich poradie si môže zvoliť používateľ zmenou čísla informačného elementu v

položke. V prípade čísla *827* nachádzajúceho sa v položke *fourth* konfiguračného súboru na obrázku 6 – 2 sa jedná o agregáciu zdrojovej IPv4 a IPv6 adresy odchyteného paketu súčasne. Keďže každá z verzií adries je protokolom IPFIX špecifikovaná ako samostatný informačný element, je nutné zadať do položky ich kombináciu. Číslo *8* reprezentuje informačný element *sourceIPv4Address* a číslo *27* informačný element *sourceIPv6Address*. Totožný princíp funguje aj pri potrebe agregácie cieľových IP adries (položka *third* sekcie *aggregation* obr.6 – 2).

6.2 Modifikácia zaraďovania paketov do tokov

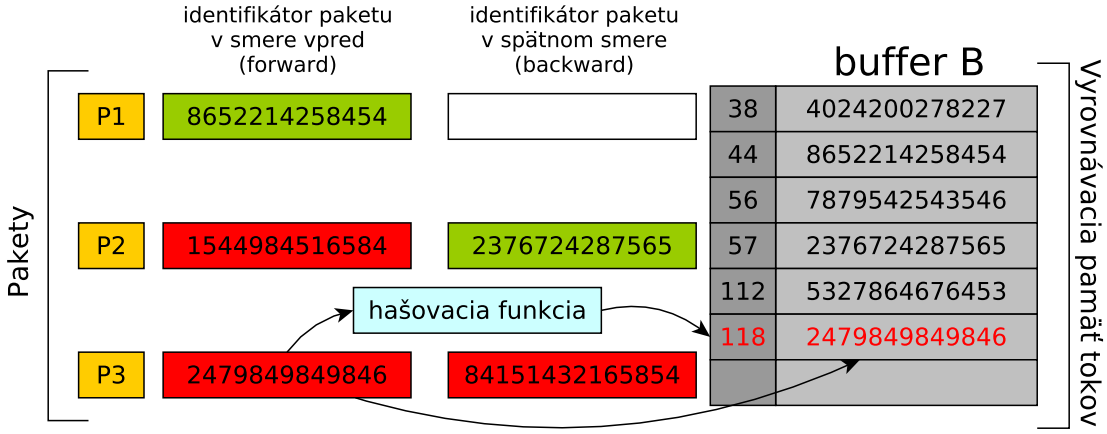
Ako už bolo v sekcii 5.2 navrhnuté, bolo potrebné modifikovať zaraďovanie paketov do tokov. V súčasnej implementácii obsahuje vyrovnávacia pamäť tokov počas behu programu záznam pre každý tok prechádzajúci meracím bodom, ktorý neexpiroval žiadnym so spôsobov a do ktorého prišiel aspoň jeden paket. Každý záznam o toku je reprezentovaný dátovou štruktúrou, pričom každá obsahuje odkaz na nasledujúcu štruktúru na danej pozícii hašovaného spájaného zoznamu. Takto je možné sa v procese identifikácie toku, do ktorého patrí daný paket, efektívne pohybovať medzi jednotlivými štruktúrami, pričom ak sa nenájde zhoda v žiadnej zo štruktúr, program vytvorí nový záznam o toku, keďže zrejme paket, ktorý sa snaží MyBeem zaradiť je prvým paketom nového zachyteného toku. Záznamy o tokoch sú pri ich vytvorení zaradené podľa zahašovania istých parametrov do hašovaného spájaného zoznamu (buffera).

Zatrieďovanie do tokov doteraz prebiehalo porovnávaním pätice kľúčových hodnôt toku. Toto správanie je potrebné ponechať v prípade vypnutého procesu agregácie a žiadnym spôsobom ho nemodifikovať. Je potrebné, aby celkový proces agregácie fungoval ako samostatný modul a žiadnym spôsobom nemodifikoval predošlé správanie programu v prípade, že nie je aktivovaný. Celý proces má najnižšiu vrstvu nástroja SLAmeter obohacovať ako samostatný výpočtový modul.

V novej implementácii zatrieďovania paketov, ak dôjde k odchyteniu paketu v me-

racom procese, daný paket musí prejsť zatrieďovacou funkciou *struct cache_item *packet_to_flow(packet_info_t *packet)*, ktorá má ako návratovú hodnotu smerník na štruktúru toku typu *cache_item*, do ktorej má byť daný paket priradený. Ešte predtým je ale nutné na základe funkcie *uint64_t fwd_packet_flow_identificator(packet_info_t *packet)*, ktorá má ako návratovú hodnotu číslo veľkosti 64 bitov, vypočítať jednoznačný identifikátor príslušnosti paketu voči toku, ten opäť zahašovať a následne získať podľa tejto hodnoty konkrétnu položku toku z vyrovnávacej pamäte tokov.
Návratová hodnota o veľkosti 64 bitov je potrebná z dôvodu, že rovnakou hašovacou funkciou musí byť vytvorený aj identifikátor toku *flowId*, ktorý je zároveň jedným z informačných elementov IPFIX sady a má predpísanú veľkosť 64 bitov, a ktorý musí byť zhodný s identifikátorom paketu.
Ak je zapnutá podpora identifikácie obojsmerných tokov *biflows* a nedošlo k zhode identifikátorov paketu a toku v smere *forward*, je vypočítaná ešte jedna hodnota funkciou *uint64_t bwd_packet_flow_identificator(packet_info_t *packet)*, aby bolo možné jednoznačne určiť, či sa nejedná o paket spätného smeru (smer *backward*). V oboch prípadoch je hodnota výsledku hašovacej funkcie porovnaná s príslušnou hodnotou identifikátora *flowId* záznamu vo vyrovnávacej pamäti. Ak nedôjde k zhode, daná funkcia vráti hodnotu 0, čo v ďalšom procese zatrieďovania znamená, že sa jedná o paket nového toku, pre ktorý je nutné vytvoriť záznam, vo vyrovnávacej pamäti tokov.

Na obr.6 – 3 si môžme všimnúť niekoľko modelových situácií pri zaraďovaní paketu do toku pri zapnutej identifikácii obojsmerných tokov *biflows*. V pravej časti obrázka je znázornená vyrovnávacia pamäť, ktorej štruktúra bude rovnaká, či už bude agregácia aktívna alebo nie.
Paket P1 je priradený do toku po nájdení zhody medzi identifikátorom paketu v smere vpred a jemu prislúchajúcim identifikátorom toku *flowId*. V tomto prípade sa jedná o paket v smere *forward direction*. Paket P2 je priradený do príslušného toku až po nájdení zhody s identifikátorom paketu v spätnom smere. Tentokrát sa jedná o paket spätného toku, teda paket v smere *backward direction*. Samozrejme



Obrázok 6 – 3: Zaraďovanie paketov do tokov pri zapnutej podpore obojsmerných tokov *biflows*

je potrebné uložiť identifikátor smeru paketu do premennej v dátovej štruktúre paketu, aby bolo možné zaradiť smerník na štruktúru paketu do správneho zoznamu v príslušnom poli toku. Paket P3 nie je zaradený do žiadneho z aktuálnych tokov, nachádzajúcich sa vo vyrovnávacej pamäti, keďže sa jedná o prvý paket daného toku a teda nenastala zhoda ani s jedným z jeho identifikátorov v oboch smeroch. Preto je paketu vytvorená položka vo vyrovnávacej pamäti tokov, kde je uložená hodnota identifikátora paketu v smere vpred, teda *forward direction*, ako identifikátor *flowId* daného toku.

6.3 Spustenie procesu agregácie

Ako už bolo skôr spomenuté, celý proces agregácie, ak je spustený, funguje v programe MyBeem v samostatnom vlákne. Prostredníctvom samostatného vlákna je možné vyhnúť sa chybám, ktoré by mohli prenechať toky, ktoré spĺňajú podmienku agregácie a majú byť agregované, exportovaciemu procesu. Toto vlákno preto pravidelne pristupuje k vyrovnávacej pamäti tokov, aby selektovalo a agregovalo všetky toky, ktoré niesú používateľom považované za rozsiahle.

K spusteniu samotného procesu agregácie môže dôjsť dvoma spôsobmi. Prvým spôsobom je, že používateľ si danú možnosť zvolí v konfiguračnom súbore *config.xml* programu MyBeem. Vtedy dôjde k spusteniu agregácie od začiatku odchytávania paketov. Druhým spôsobom je automatické spustenie agregácie. K tomu, aby k nej došlo, je nutné vyplniť príslušnú premennú v konfiguračnom súbore (obr. 6 – 2). Pri automatickej agregácii záznamov o tokoch dochádza pred vytvorením záznamu o novom toku vo funkcii *void add_packet_to_flow(struct cache_item *item, struct _packet_info_t *packet)* ku kontrole vyťaženia vyrovnávacej pamäte tokov, kde sa prostredníctvom porovnania premennej *FLOW_CACHE_SIZE*, ktorá reprezentuje veľkosť alokovanej vyrovnávacej pamäte, a návratovej hodnoty funkcie *list_size((struct list *)&cache_free_list)* stanoví aktuálne percentuálne vyťaženie, aby tak bolo zabezpečené automatické spustenie agregácie pri vyťažení vyrovnávacej pamäte na viac ako 75% jej celkového rozsahu.
Inou možnosťou by bolo upraviť správanie sa vyrovnávacej pamäte tak, aby bolo možné stále pri určitom percentuálnom vyťažení dynamicky realokovať množstvo vyhradenej pamäte. Tento spôsob by však mohol so sebou priniesť nemalé komplikácie v podobe celkového vyťaženia operačnej pamäte stroja, keďže generovaním dostatočného množstva rôznych tokov by k realokácii a zväčšovaniu pamäte dochádzalo veľmi často, až pokiaľ by sa neprečerpala dostupná pamäť stroja, na ktorom je spustený MyBeem.

6.4 Vytvorenie zoznamov pre agregované toky a selekcia tokov určených pre agregáciu

Ak dochádza k agregácii, je potrebné pre každú skupinu agregovaných tokov (skupinou sa myslí súbor tokov, ktoré majú kľúčové hodnoty redukované o ten istý počet hodnôt), tak ako je to popísané v kapitole 5.6, vytvoriť hašovaný spájaný zoznam (buffer). Hash list resp. hašovaný spájaný zoznam je dátová štruktúra, do ktorej

je hodnota vložená na základe k nej vypočítanej hodnoty hašovaného atribútu. V prípade, že je viacero hodnôt zahašovaných na to isté číslo, a teda sú v hašovanom zozname uložené pod tou istou referenciou, sú tieto prvky prepojené prostredníctvom smerníka na nasledujúci prvok v atribúte *struct cache_item *next*. Preto boli v programe MyBeem pre účely agregačného procesu vytoverené 4 dátové štruktúry tohto typu. Ak je tok agregovaný a v bufferi, do ktorého bude presunutý sa nenachádza tok so shodným identifikátorom *flowId*, nenakopírujú sa všetky atribúty tohto toku do novej štruktúry typu *struct cache_item*, ale do vytvoreného hašovaného zoznamu je skopírovaný len smerník na danú štruktúru a samozrejme je táto štruktúra agregovaná o príslušný počet kľúčových hodnôt, podľa toho o aký stupeň agregácie sa v danom prípade jedná. Všetky tieto hašované zoznamy spoločne tvoria vyrovnávaciu pamäť tokov programu MyBeem.

Testovaním prevádzky na serveri laboratória CNL Technickej univerzity v Košiciach bolo zistené, že v priemere je paket spracovaný programom MyBeem každých 35000 nanosekúnd a po zohľadnení veľkosti primárne alokovanej pamäte na hodnotu 8192 bajtov, bol časový interval kontroly vyrovnávacej pamäte tokov primárne stanovený na 250 milisekúnd. Táto hodnota reprezentuje veličinu spúšťač agregácie (*aggregation trigger* A_t) popísanú v sekcii 5.4. Za tento čas je program v priemere schopný spracovať približne 7150 paketov. Taktiež bolo meraniami v reálnej sieťovej prevádzke na serveri laboratória CNL Technickej univerzity v Košiciach zistené, že takmer 92% zachytených tokov nemá hodnotu veličiny *octet total count* väčšiu ako 10000(tab. 6 – 1). Teda MyBeem nezachytil po dobu pasívnej expirácie toku viac oktetov patriacich danému toku.

S ohľadom na pokusné merania bola teda hodnota veličiny *aggregation condition* A_c, ktorá je hraničnou a určujúcou hodnotou pre agregáciu daného toku (bližšie popísaná v sekcii 5.5), primárne stanovená na 10000 oktetov v danom toku. Teda ak v toku po dobu uplynutia časového intervalu *aggregation trigger* A_t nebude zachytených viac ako 10000 oktetov (0,01 Mb), tok bude agregovaný. Ako už bolo spomenuté, tieto hodnoty sú východzie, teda sú takto primárne nastavené po stiahnutí programu. Sa-

Tabuľka 6 – 1: Namerané hodnoty počtu oktetov v tokoch v reálnej sieťovej prevádzke

Počet oktetov v toku	Toky s daným počtom oktetov	Celkový počet tokov
viac ako 10000	814031	11021322
viac ako 50000	545451	11021322
viac ako 100000	482043	11021322

mozrejmosťou je možnosť nastavenia týchto veličín používateľom v konfiguračnom súbore, aby ich bolo možné prispôsobiť pre merania v rôznych typoch sietí.

6.5 Redukcia kľúčových hodnôt toku

Redukcia kľúčových hodnôt toku sa odohráva vo funkcii *void flow_keys_reduction (struct cache_item *item, int buff_nmbr)*, kde je ako parameter zasielaný smerník na konkrétnu štruktúru *cache_item*, ktorej kľúčové hodnoty je potrebné redukovať a taktiež číslo buffera, v ktorom sa daný tok nachádza. Ako už bolo spomenuté, kľúčové hodnoty toku, reprezentované číslami informačných elementov, boli v poradí, v akom budú na jednotlivých agregačných úrovniach redukované, nahraté do premenných programu pri jeho inicializácii. Funkcia redukcie kľúčových hodnôt tieto údaje číta a v závislosti od čísla v premennej *buff_nmbr* ich následne redukuje. Do funkcie stále vstupuje aktuálne číslo buffera, v ktorom sa daný tok nachádza, a teda je potrebné redukovať tok stále o taký počet kľúčových hodnôt, ktorý je od tohto čísla o jednotku väčší, keďže tok bude v procese agregácie zaradený do nasledujúcej agregačnej úrovne.

6.6 Presúvanie tokov medzi jednotlivými bufframi a ich agregácia

K samotnému presúvaniu tokov medzi jednotlivými buffermi dochádza vo funkcii *void flow_cache_rework(void)*, ktorá je volaná stále v pravidelných intervaloch zo sekcie kódu nachádzajúcej sa vo vlákne pre agregáciu tokov. Zároveň je potrebné spomenúť, že nieje presúvaný celý tok, v zmysle prekopírovania jeho údajov do novej štruktúry vyrovnávacej pamäte, ale je presunutý iba smerník na položku typu *struct cache_item* do nového, na to určeného buffera.

Vo tejto funkcii sa v cykle prejdú všetky položky jednotlivých zoznamov a z nich sa vyberú záznamy o tokoch, ktoré nevyhovujú podmienke o minimálnom počte oktetov v toku.

Každá položka je zasielaná ako parameter funkcie *void move_flows_to_next_buff(struct cache_item *item, int i, int buff_nmbr)* spolu s číslom položky v bufferi *i* a číslom buffera *buff_nmbr*, do ktorého budeme danú položku presúvať. Následne je z tejto funkcie volaná funkcia na redukciu kľúčových hodnôt toku *void flow_keys_reduction (struct cache_item *item, int buff_nmbr)*. Po redukovaní kľúčových hodnôt toku je vypočítaná nová hodnota identifikátora toku *flowId*. Potom je potrebné zistiť, či sa už v bufferi, do ktorého chceme uložiť redukovaný tok, nenachádza tok s rovnakou hodnotou identifikátora *flowId*. Ak nie, je potrebné odstrániť z pôvodného buffera smerník na tento redukovaný tok a premiestniť ho do tohto prehľadávaného buffera na index podľa zahašovania hodnoty identifikátora *flowid*. Zároveň je potrebné vymazať zoznami paketov prislúchajúcich danému toku a na miesto každého odstráneného paketu nahrať do vyrovnávacej pamäte paketov *packet cache* prázdnu štruktúru *struct list pcache_free_list*. Tieto zoznami slúžia pre meranie charakteristík jednosmerného oneskorenia OWD, ktoré bude takto možné merať len pri vypnutom procese agregácie.

Ak sa ale nájde položka s rovnakou hodnotou *flowId*, akú ma redukovaný tok, ktorý chceme do buffera umiestniť, musí dôjsť k zlúčeniu týchto dvoch tokov a spočíta-

niu príslušných polí funkciou *void merge_flows(struct cache_item *from, struct cache_item *to)*.

Samotné spájanie tokov, a teda aj aktualizácia záznamov o tokoch v procese agregácie v jednotlivých bufferoch sa odohráva vo funkcii *void merge_flows(struct cache_item *from, struct cache_item *to)*. Do tejto funkcie sú ako parametre zasielané smerníky na dvojicu štruktúr, z ktorých jedna (štruktúra *from*) obsahuje záznam o toku, ktorý má po agregácii o príslušnú hodnotu rovnaký identifikátor toku ako záznam *to*, a teda je potrebné ho s týmto už exitujúcim záznamom spojiť. Postupne sú preto aktualizované jednotlivé položky štruktúry *to* na základe hodnôt záznamu *from*. Taktiež je nutné aktualizovať vyrovnávaciu pamäť paketov *packet cache* odstránením paketov prislúchjúcich danému toku zo zoznamov, ktoré sú položkami daného záznamu o toku *from*. Na ich miesto je potrebné nahrať prázdnu štruktúru *struct list pcache_free_list*, do ktorej bude možné v ďalšom procese uložiť nové záznamy o odchytených paketoch.

Tento proces je nutné vykonať z dôvodu možného postupného preplnenia vyrovnávacej pamäte paketov *packet cache*. Pretože ak by dochádzalo vo veľkej miere len k aktívnej expirácii agregovaných tokov, záznamy by neboli zahodené, keďže pri aktívnej expirácii toku je exportovaná len kópia záznamu o toku a tým by sa vo vyrovnávacej pamäti paketov mohlo veľmi rýchlo nahromadiť množstvo paketov prekračujúce veľkosť alokovanej pamäte pre odchytené pakety *packet cache*, čo by mohlo viesť až k úplnému zlyhaniu meracieho procesu. Záznam *from* je po ukončení tohto procesu vymazaný a na jeho miesto je nahraná prázdna štruktúra *struct list cache_free_list*, aby bolo možné na túto pozíciu uložiť v ďalšom procese odchytávania paketov záznam o novom toku.

Tok môže byť v jednom agregačnom cykle preusnutý postupne o niekoľko agregačných úrovní (ak je agregovaný do iného toku), keďže sa buffere prehľadávajú postupne od základného, v ktorom sú neagregované záznamy o tokoch, až po ten, kde sú toky agregované o 4 kľúčové hodnoty. Takto dôjde v jednom agregačnom cykle k úplnému preusporiadaniu vyrovnávacej pamäte.

6.7 Časová expirácia agregovaných a neagregovaných tokov

Nad každým z bufferov pre agregované aj neagregované toky zároveň prebieha kontorla vlákna pre časovú expiráciu tokov, ktorá sleduje, či v prípade daných tokov nenastala pasívna alebo aktívna expirácia, alebo či nebol zachytený FIN paket daného toku. Kontrola zachytenia koncového FIN paketu toku prebieha len v prvom bufferi, kde nie sú toky agregované, pretože zachytenie FIN paketu v agregovanom toku môže znamenať iba ukončenie jedného z tokov, ktoré sú do daného toku agregované a nie aj ukončenie celého agregovaného toku.

Ak by k toku, ktorý expiroval z dôvodu zachytenia FIN paketu pristúpilo ako prvé vlákno pre expiráciu tokov ešte pred uplynutím intervalu pre spustenie agregačného procesu *aggregation trigger* A_t, daný tok by bol exportovaný bez akejkoľvek agregácie aj napriek tomu, že by nemusel spĺňať agregačnú podmienku minimálneho počtu oktetov toku *aggregation condition* A_c. Preto bolo nutné vo vlákne pre expiráciu tokov, pri expirácii toku zachytením FIN paketu skontrolovať, či daný tok spĺňa agregačnú podmienku a ak nie, agregovať daný záznam o toku, aby nedošlo k chybnému exportu.

Ak teda tok v niektorom z bufferov expiruje niektorým zo spôsobov, je prenechaný exportovaciemu procesu a údaje o ňom sú zaslané zhromažďovaciemu procesu nástroja SLAmeter.

Odosielanie informácií o agregovaných tokoch prebieha prostredníctvom IPFIX správ, ku ktorým sú priradené samostané šablóny. To znamená, že toky redukované o ten istý počet kľúčových hodnôt budú mať v procese zasielnania údajov zhromažďovaciemu procesu svoju samostatnú šablónu, z ktorej budú odobrané polia práve tých kľúčových hodnôt toku, o ktoré je daný tok redukovaný. Zhromažďovací proces takto bude schopný rozoznať jednotlivé záznami o agregovanom toku v IPFIX správe a dekódovať ich na základe uvedeného čísla šablóny.

6.8 Implementácia informačných elementov súvisiacich s agregáciou

Keďže je nutné vyvíjať program MyBeem v súlade s normami IPFIX protokolu, je potrebné implementovať aj niekoľko informačných elementov, ktoré budú slúžiť na zdokonalenie práce s agregovanými tokmi a zároveň používateľovi poskytnú detailnejšie informácie o týchto tokoch (tab.6 – 2).

Tabuľka 6 – 2: Tabuľka informačných elementov používaných pri agregácii tokov

Číslo informačného elementu	Názov informačného elementu
375	originalFlowsPresent
376	originalFlowsInitiated
377	originalFlowsCompleted
378	distinctCountOfSourceIPAddress
379	distinctCountOfDestinationIPAddress
380	distinctCountOfSourceIPv4Address
381	distinctCountOfDestinationIPv4Address
382	distinctCountOfSourceIPv6Address
383	distinctCountOfDestinationIPv6Address

- ***originalFlowsPresent*** - reprezentuje počet originálnych tokov, ktoré boli agregované do daného agregovaného toku.
- ***originalFlowsInitiated*** - reprezentuje počet originálnych tokov, ktorých prvý (inicializačný) paket sa nachádza v danom agregovanom toku.
- ***originalFlowsCompleted*** - reprezentuje počet originálnych tokov, ktorých koncový (FIN) paket sa nachádza v danom agregovanom toku.
- ***distinctCountOfSourceIPAddress*** - počítadlo jedinečných zdrojových IP adries originálnych tokov, ktoré boli agregované do daného toku, bez ohľadu

na verziu IP protokolu. Tento informačný element je potrebné preferovať pred informačnými elementami, ktoré špecifikujú aj verziu IP protokolu v počítadle, pokiaľ nieje potrebné oddeľovať počty tokov jednotlivých verzií.

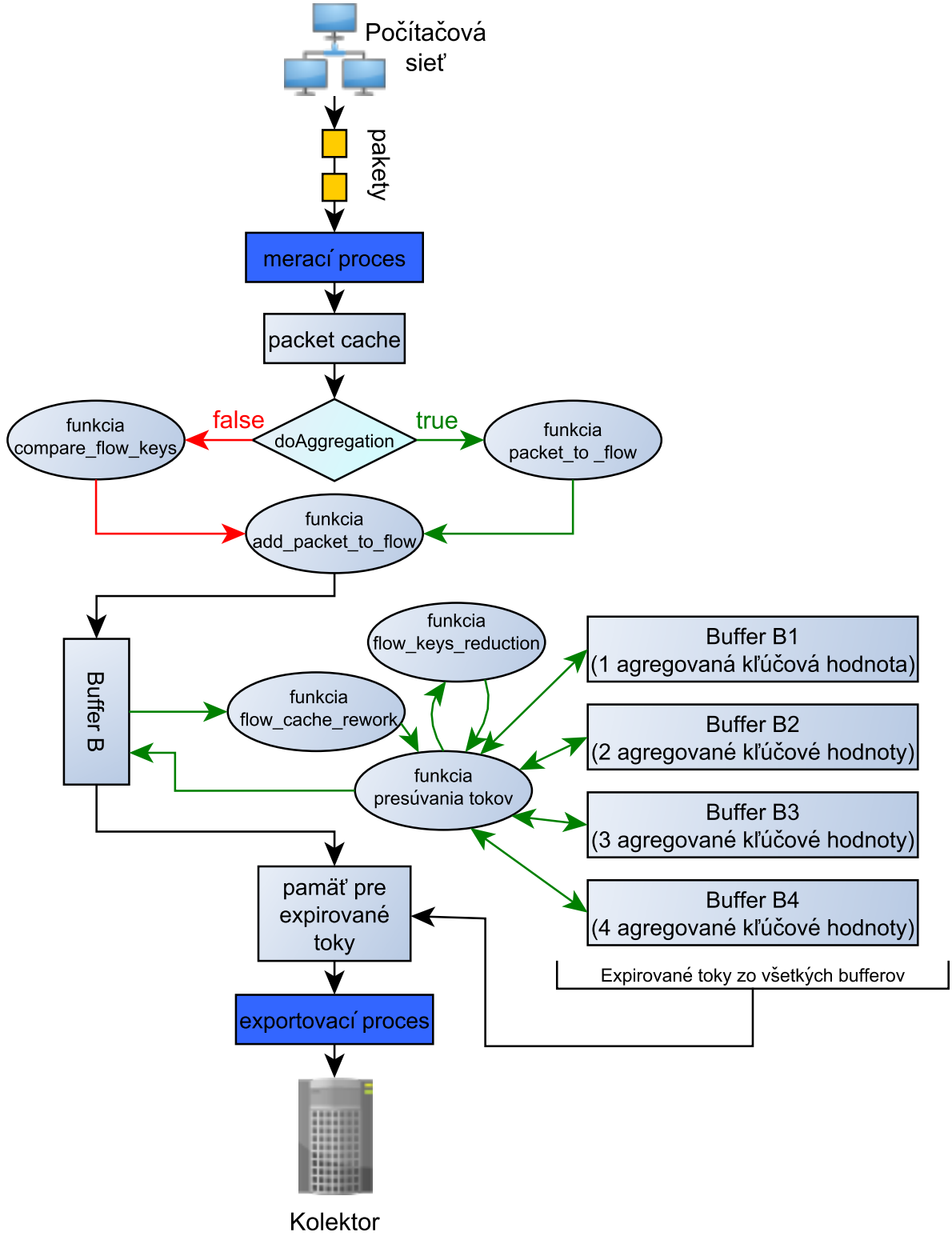
- ***distinctCountOfDestinationIPAddress*** - počítadlo jedinečných cieľových IP adries originálnych tokov, ktoré boli agregované do daného toku, bez ohľadu na verziu IP protokolu. Pre tento element platia priority podobne ako pre *distinctCountOfSourceIPAddress*.
- ***distinctCountOfSourceIPv4Address*** - počítadlo originálnych zdrojových IPv4 adries, ktoré sa podieľajú na danom agregovanom toku.
- ***distinctCountOfDestinationIPv4Address*** - počítadlo originálnych cieľových IPv4 adries, ktoré sa podieľajú na danom agregovanom toku.
- ***distinctCountOfSourceIPv6Address*** - počítadlo originálnych zdrojových IPv6 adries, ktoré sa podieľajú na danom agregovanom toku.
- ***distinctCountOfDestinationIPv6Address*** - počítadlo originálnych cieľových IPv6 adries, ktoré sa podieľajú na danom agregovanom toku.

6.9 Celkový popis procesu spracovania údajov v programe MyBeem

Obrázok 6 – 4 znázorňuje súčasný stav implementácie spracovania údajov o tokoch v programe Mybeem. Z celkového hľadiska, ak dochádza k agregácii údajov, je tento proces do značnej miery modifikovaný v porovnaní s pôvodným procesom spracovania. Merací proces (bližšie popísaný v sekcii 2.1) odchytáva pakety z počítačovej siete a posiela ich na spracovanie programu. Pri vypnutej funkcii agregácie tokov sú pakety zatrieďované do tokov porovnávaním pätice ukazovateľov funkciou *int compare_keys(struct flow_key *fk, packet_info_t *pkt)*. Z vyrovnávacej pamäte tokov (buffer B) sú toky po ich expirácii presunuté do pamäte expirovaných tokov a ná-

sledne sú exportované na kolektor. Ak je proces agregácie zapnutý v programe sú prítomné dva základné procesy, čo sa týka spravovania vyrovnávacej pamäte tokov:

- **proces zaraďovania paketov do tokov prostredníctvom identifikátorov paketov** - zaraďovanie paketov do tokov funguje v programe stále na úrovni neagregovaných záznamov o tokoch. Nemôže sa teda stať, aby bol paket zaradený funkciou do buffera s agregovanými hodnotami záznamov o tokoch. V prvotnej implementácii bola testovaná aj možnosť zaraďovania paketov do agregovaných tokov. Skúmaním log súboru programu sa však zistilo, že MyBeem často krát zaradil pakety rozsiahlych tokov do agregovaných záznamov, pretože nenašiel žiadny vyhovujúci záznam v neagregovanom bufferi. Týmto spôsobom dochádzalo k stratám údajov o rozsiahlych tokoch. Samotný proces zatrieďovania teda prebieha vo funkcii *add_packet_to_flow*, ktorá využíva návratovú hodnotu funkcie *packet_to_flow*(obr.6 – 4)
- **proces agregácie záznamov o tokoch** - ak záznam o toku splní podmienku agregácie *aggregation condition* A_c bude po redukovaní jeho kľúčových hodnôt prostredníctvom funkcie presúvania tokov umiestnený do jedného z bufferov a prípadne aj agregovaný funkciou *merge_flows* so záznamom o toku, ktorý ma v danom bufferi s ním zhodný identifikátor toku *flowId*.



Obrázok 6 – 4: Grafický popis procesu spracovania údajov v programe MyBeem

7 Overenie implementácie modulu pre agregáciu tokov

Overenie implementácie modulu agregácie v programe MyBeem prebiehalo v reálnej, ale aj v umelo generovanej sieťovej prevádzke. Generácia prevádzky bola nevyhnutná najmä z dôvodu úspešného odtestovania jednotlivých reakcií na špecifické modelové situácie, ktorých testovanie by sa v reálnej sieťovej prevádzke dosahovalo len veľmi ťažko, keďže výskyt týchto situácií je v sieti ojedinelý, ale napriek tomu závažný pre monitorovaciu aplikáciu.

Na generovanie sieťovej prevádzky bol použitý generátor IP paketov Mausezahn. Tento program je rýchlym paketovým generátorom napísaným v programovacom jazyku C a fungujúcim ako konzolová aplikácia v operačnom prostredí Ubuntu. Umožňuje zasielať veľké množstvá paketov na dané monitorované rozhranie a tým efektívne odtestovať novoimplementovanú fnkcionalitu programu MyBeem. Zároveň je tu možné generovať pakety s náhodne zvolenou zdrojovou IP adresou a tým zabezpečiť vznik veľkého množstva krátkych tokov, teda jednu z modelových situácií, ktoré sa snaží riešiť táto diplomová práca.

Pre lepšiu názornosť funkcionality procesu agregácie bolo v pokusoch zvolené poradie redukcie kľúčových hondnôt toku na základe tabuľky 7 – 1.

V starej verzii programu MyBeem pri zachytení veľkého množstva paketov, ktoré vytvorili veľké množstvo tokov, nastával problém preplnenia vyrovnávacej pamäte tokov *flow_cache* a následná celková strata informácii o zachytávanej sieťovej prevádzke. Preto bolo pre demonštráciu funkčnosti riešenia programom Mausezahn vygenerovaných 10 000 paketov (obr. 7 – 1) s rôznou hodnotou atribútu *source-Ipv4address*

Veľkosť vyrovnávacej pamäte tokov programu MyBeem bola nastavená na hodnotu 8192 bajtov. Z daného údaju vyplíva, že program je schopný udržať bez aktivovaného agregačného procesu maximálne 8192 rôznych tokov v danom čase, pokiaľ

Tabuľka 7 – 1: Poradie redukcie kľúčových hodnôt toku pri testovaní procesu agregácie v programe MyBeem

Informačné elementy slúžiace ako kľúčové hodnoty	Priorita kľúčových hodnôt
protocolIdentifier	1 (najvyššia)
sourceTransportPort	2
destinationTransportPort	3
destinationIPv4Address/destinationIPv6Address	4
sourceIPv4Address/sourceIPv6Address	5 (najnižšia)

nedôjde k ich pasívnej expirácii.

Na obrázku 7 – 2 je zobrazená reakcia programu na zachytenie nadmerného množ-

```
unclesam@ubuntu:~$ sudo mz lo -d 2000 -t udp "sp=2222,dp=3333"
 -c 10000 -A rand -B 127.0.0.1
Mausezahn will send 10000 frames... 0.44 seconds (22727 packet
s per second)
```

Obrázok 7 – 1: Zaslanie 10000 paketov programom Mausezahn

stva novovzniknutých tokov bez aktivovaného procesu agregácie. Hoci program do istého času úspešne spracovával zachytené pakety, od istého momentu došlo k preťaženiu vyrovnávacej pamäte tokov a k chybovému výpisu *Flow cache full*, pri ktorom sa program snaží dostať zo vzniknutej kritickej situácie tým, že nespracováva, ale zahadzuje odchytené pakety až do momentu, kým sa mu neuvoľní aspoň jedna štruktúra pre uloženie záznamu o toku, čím ale dochádza k masívnej strate monitorovanej informácie, najmä ak pretrváva výskyt veľkého množstva krátkych novozachytených tokov (napr. typ DoS útoku na zariadenie nachádzajúce sa v sieti za meracím bodom).

Naopak, pri aktivovanom module agregácie a pri rovnakých podmienkach v počítačovej sieti, kedy bolo vygenerovaných 10 000 nových tokov, je možné sledovať, že

```
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
29.4.2014  23:33:5  root  Beem  ERROR  Flow Cache Full
```

Obrázok 7 – 2: Preplnenie vyrovnávacej pamäte tokov pri deaktivovanej agregácii tokov

program si s touto záťažou úspešne poradil a bol schopný nepretržitej prevádzky, bez akéhokoľvek výpadku v spracovávaní zachytených paketov (obr.7 – 3). Odchytenie daného množstva tokov si po ich agregácii vyžiada obsadenie iba jednej štruktúry vo vyrovnávacej pamäti tokov (samozrejme po redukovaní kľúčovej hodnoty *sourceIPv4Address*) namiesto 10 000 štruktúr pri vypnutom procese agregácie.

```
29.4.2014  23:23:6  root  Beem  INFORMATION  Expiration reason value -> 1
29.4.2014  23:23:6  root  Beem  INFORMATION  Octet total count -> 280000
29.4.2014  23:23:6  root  Beem  INFORMATION  Type of used protocol -> [UDP]
29.4.2014  23:23:6  root  Beem  INFORMATION  Source IP address ->
29.4.2014  23:23:6  root  Beem  INFORMATION  Destination IP address -> 127.0.0.1
29.4.2014  23:23:6  root  Beem  INFORMATION  Packet total count -> 10000
29.4.2014  23:23:6  root  Beem  INFORMATION  Source port -> 2222
29.4.2014  23:23:6  root  Beem  INFORMATION  Destination port -> 3333
29.4.2014  23:23:6  root  Beem  INFORMATION  Speed -> 8 kBps
29.4.2014  23:23:6  root  Beem  INFORMATION  Amount of DATA sent -> 279888:0
29.4.2014  23:23:6  root  Beem  INFORMATION  EXPORTED FROM -> 1
```

Obrázok 7 – 3: Reakcia programu pri zachytení 10000 novovytvorených tokov pri aktivovanej agregácii tokov

7.1 Príklady výstupov programu pri exporte agregovaných tokov

Na nasledujúcich obrázkoch sú znázornené exporty tokov z jednotlivých bufferov, kde sú toky agregované o rôzny počet kľúčových hodnôt.

```
30.4.2014  0:11:53  root  Beem  INFORMATION  Flow_ID value: 13790327740199059789
30.4.2014  0:11:53  root  Beem  INFORMATION  Expiration reason value -> 2
30.4.2014  0:11:53  root  Beem  INFORMATION  Octet total count -> 124
30.4.2014  0:11:53  root  Beem  INFORMATION  Type of used protocol -> [UDP]
30.4.2014  0:11:53  root  Beem  INFORMATION  Source IP address ->
30.4.2014  0:11:53  root  Beem  INFORMATION  Destination IP address ->
30.4.2014  0:11:53  root  Beem  INFORMATION  Packet total count -> 2
30.4.2014  0:11:53  root  Beem  INFORMATION  Source port -> 21014
30.4.2014  0:11:53  root  Beem  INFORMATION  Destination port -> 53
30.4.2014  0:11:53  root  Beem  INFORMATION  Speed -> 15692 kBps
30.4.2014  0:11:53  root  Beem  INFORMATION  Amount of DATA sent -> 124:0
30.4.2014  0:11:53  root  Beem  INFORMATION  EXPORTED FROM -> 2
```

Obrázok 7 – 4: Export toku redukovaného o 2 kľúčové hodnoty

```
30.4.2014  0:11:56  root  Beem  INFORMATION  Flow_ID value: 5076871039819262555
30.4.2014  0:11:56  root  Beem  INFORMATION  Expiration reason value -> 2
30.4.2014  0:11:56  root  Beem  INFORMATION  Octet total count -> 318
30.4.2014  0:11:56  root  Beem  INFORMATION  Type of used protocol -> [UDP]
30.4.2014  0:11:56  root  Beem  INFORMATION  Source IP address ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Destination IP address ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Packet total count -> 10
30.4.2014  0:11:56  root  Beem  INFORMATION  Source port ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Destination port -> 5355
30.4.2014  0:11:56  root  Beem  INFORMATION  Speed -> 0 kBps
30.4.2014  0:11:56  root  Beem  INFORMATION  Amount of DATA sent -> 81408:0
30.4.2014  0:11:56  root  Beem  INFORMATION  EXPORTED FROM -> 3
```

Obrázok 7 – 5: Export toku redukovaného o 3 kľúčové hodnoty

```
30.4.2014  0:11:56  root  Beem  INFORMATION  Flow ID value: 210653368347
30.4.2014  0:11:56  root  Beem  INFORMATION  Expiration reason value -> 2
30.4.2014  0:11:56  root  Beem  INFORMATION  Octet total count -> 2144
30.4.2014  0:11:56  root  Beem  INFORMATION  Type of used protocol -> [TCP]
30.4.2014  0:11:56  root  Beem  INFORMATION  Source IP address ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Destination IP address ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Packet total count -> 38
30.4.2014  0:11:56  root  Beem  INFORMATION  Source port ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Destination port ->
30.4.2014  0:11:56  root  Beem  INFORMATION  Speed -> 0 kBps
30.4.2014  0:11:56  root  Beem  INFORMATION  Amount of DATA sent -> 52:0
30.4.2014  0:11:56  root  Beem  INFORMATION  EXPORTED FROM -> 4
```

Obrázok 7 – 6: Export toku redukovaného o 4 kľúčové hodnoty

7.2 Overenie reakcie programu na nastavenie agregačnej podmienky

V tomto prípade bolo úlohou overiť reakciu programu MyBeem na špecifické nastavenie agregačnej podmienky. Táto podmienka bola nastavená na hodnotu 100 oktetov v toku, teda všetky toky, ktoré obsahujú v ukazovateli *octetTotalCount* číslo menšie ako 100, je nutné podrobiť procesu agregácie a toky, ktoré nesplnia danú podmienku je nutné ponechať v pôvodnom bufferi bez akéhokoľvek zásahu do záznamu.
Preto bola aj v tomto prípade programom Mausezahn vygenerovaná testovacia sieťová prevádzka. Najprv bolo na port monitorovaný meracím procesom nástroja MyBeem zaslaných 10 paketov s náhodnou zdrojovou IP adresou. Každý paket teda vytvoril vo vyrovnávacej pamäti samostatný záznam o toku, ktorý mal veľkosť ukazovateľa *octetTotalCount* rovnú 28 oktetom. Následne na to bolo na ten istý port zaslaných 10 paketov s rovnakou zdrojovou, cieľovou IP adresou, rovnakými číslami zdrojových a cieľových portov a rovnakým typom protokolu (obr.7 – 7). Dané pakety teda vo vyrovnávacej pamäti vytvorili jeden záznam o toku, ktorý je možné na základe stanovenej agregačnej podmienky považovať za rozsiahly.
V reakcii programu MyBeem na obrázku 7 – 8 si môžme všimnúť, že program rea-

```
unclesam@ubuntu:~$ sudo mz lo -d 200 -t udp "sp=2222,dp=3333"
-c 10 -A rand -B 127.0.0.1
Mausezahn will send 10 frames...
unclesam@ubuntu:~$ sudo mz lo -d 200 -t udp "sp=2222,dp=3333"
-c 10 -A 192.168.1.2 -B 127.0.0.1
Mausezahn will send 10 frames...
```

Obrázok 7 – 7: Zaslanie dvoch sledovaných skupín v počte 10 paketov programom Mausezahn

goval na vzniknutú udalosť korektne. Prvých 10 paketov zaslaných na port s rôznou zdrojovou IP adresou bolo agregovaných do jedného toku, v ktorom sa ukazovateľ počtu paketov toku *Packet total count* rovná číslu 10 (kumulovaná hodnota zo 10

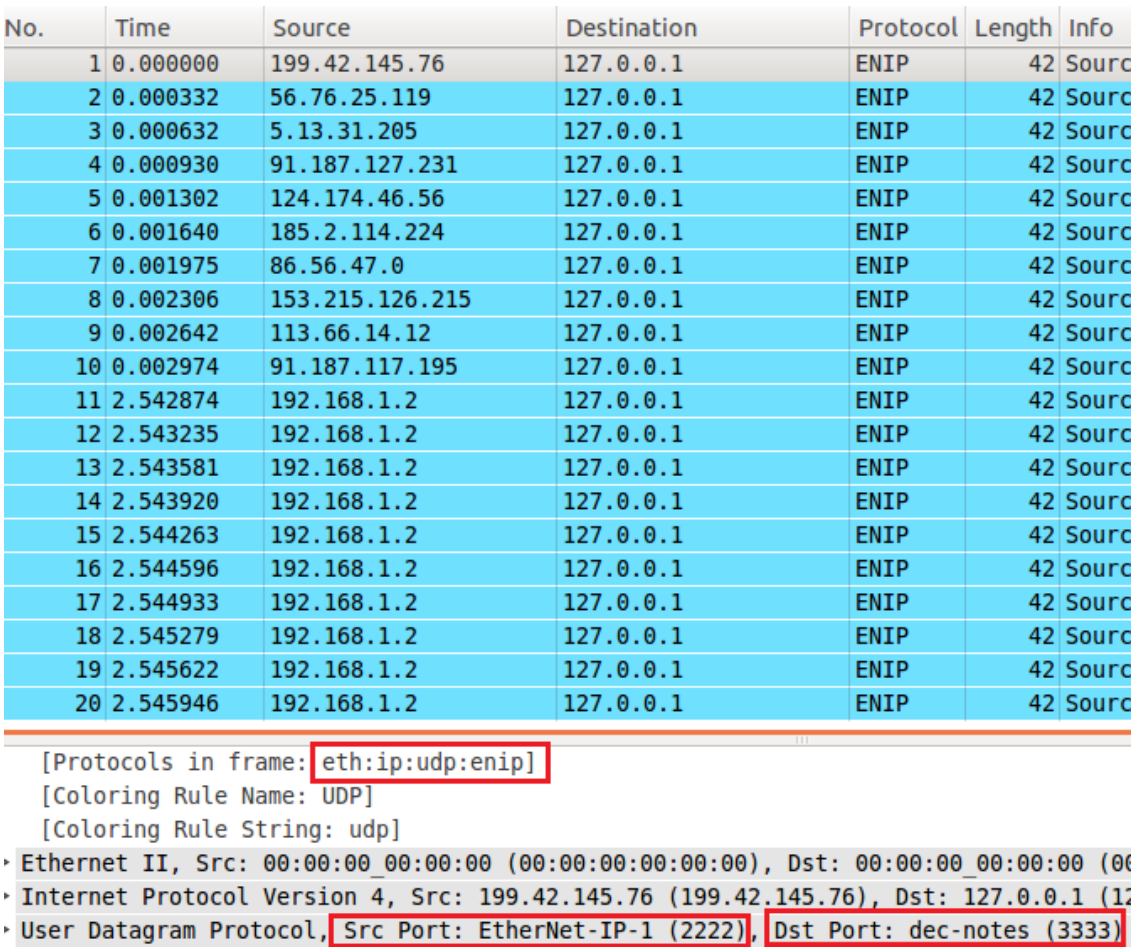
tokov po 1 pakete), pričom bol agregovaný atribút *sourceIpAddress*. V nasledujúcom zázname na obrázku 7 – 8 si môžme všimnúť druhú skupinu paketov s konštantnými hodnotami kľúčových atribútov toku. Táto skupina bola zachytená a spracovaná korektne ako jeden tok a následne exportovaná bez agregácie niektorej z kľúčových hodnôt. Položky výpisu *EXPORTED FROM* udávajú číslo buffera, z ktorého boli toky exportované. Číslo 1 znamená buffer s jednou agregovanou kľúčovou hodnotou, číslo 0 znamená, že na danom zázname nebola vykonaná agregácia.

Tento pokus bol overený aj programom Wireshark, ktorého výpis je na obrázku

```
29.4.2014  23:45:39  root  Beem  INFORMATION  Expiration reason value -> 1
29.4.2014  23:45:39  root  Beem  INFORMATION  Octet total count -> 280
29.4.2014  23:45:39  root  Beem  INFORMATION  Type of used protocol -> [UDP]
29.4.2014  23:45:39  root  Beem  INFORMATION  Source IP address ->
29.4.2014  23:45:39  root  Beem  INFORMATION  Destination IP address -> 127.0.0.1
29.4.2014  23:45:39  root  Beem  INFORMATION  Packet total count -> 10
29.4.2014  23:45:39  root  Beem  INFORMATION  Source port -> 2222
29.4.2014  23:45:39  root  Beem  INFORMATION  Destination port -> 3333
29.4.2014  23:45:39  root  Beem  INFORMATION  Speed -> 266 kBps
29.4.2014  23:45:39  root  Beem  INFORMATION  Amount of DATA sent -> 280:0
29.4.2014  23:45:39  root  Beem  INFORMATION  EXPORTED FROM -> 1
================================================================
29.4.2014  23:45:39  root  Beem  INFORMATION  Flow_ID value: 2560896617854536548
29.4.2014  23:45:39  root  Beem  INFORMATION  Expiration reason value -> 1
29.4.2014  23:45:39  root  Beem  INFORMATION  Octet total count -> 280
29.4.2014  23:45:39  root  Beem  INFORMATION  Type of used protocol -> [UDP]
29.4.2014  23:45:39  root  Beem  INFORMATION  Source IP address -> 192.168.1.2
29.4.2014  23:45:39  root  Beem  INFORMATION  Destination IP address -> 127.0.0.1
29.4.2014  23:45:39  root  Beem  INFORMATION  Packet total count -> 10
29.4.2014  23:45:39  root  Beem  INFORMATION  Source port -> 2222
29.4.2014  23:45:39  root  Beem  INFORMATION  Destination port -> 3333
29.4.2014  23:45:39  root  Beem  INFORMATION  Speed -> 13 kBps
29.4.2014  23:45:39  root  Beem  INFORMATION  Amount of DATA sent -> 280:0
29.4.2014  23:45:39  root  Beem  INFORMATION  EXPORTED FROM -> 0
```

Obrázok 7 – 8: Reakcia programu pri zachytení dvoch analyzovaných skupín paketov

7 – 9. Rozdiel vo výpise je jedine v type protokolu. *ENIP* je protokol aplikačnej vrstvy, do ktorého generátor paketov Mausezahn zapúzdril protokol UDP. MyBeem detekoval protokol UDP, pretože v meracom procese nie je doposiaľ implementované rozlišovanie protokolov aplikačnej vrstvy.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	199.42.145.76	127.0.0.1	ENIP	42	Sourc
2	0.000332	56.76.25.119	127.0.0.1	ENIP	42	Sourc
3	0.000632	5.13.31.205	127.0.0.1	ENIP	42	Sourc
4	0.000930	91.187.127.231	127.0.0.1	ENIP	42	Sourc
5	0.001302	124.174.46.56	127.0.0.1	ENIP	42	Sourc
6	0.001640	185.2.114.224	127.0.0.1	ENIP	42	Sourc
7	0.001975	86.56.47.0	127.0.0.1	ENIP	42	Sourc
8	0.002306	153.215.126.215	127.0.0.1	ENIP	42	Sourc
9	0.002642	113.66.14.12	127.0.0.1	ENIP	42	Sourc
10	0.002974	91.187.117.195	127.0.0.1	ENIP	42	Sourc
11	2.542874	192.168.1.2	127.0.0.1	ENIP	42	Sourc
12	2.543235	192.168.1.2	127.0.0.1	ENIP	42	Sourc
13	2.543581	192.168.1.2	127.0.0.1	ENIP	42	Sourc
14	2.543920	192.168.1.2	127.0.0.1	ENIP	42	Sourc
15	2.544263	192.168.1.2	127.0.0.1	ENIP	42	Sourc
16	2.544596	192.168.1.2	127.0.0.1	ENIP	42	Sourc
17	2.544933	192.168.1.2	127.0.0.1	ENIP	42	Sourc
18	2.545279	192.168.1.2	127.0.0.1	ENIP	42	Sourc
19	2.545622	192.168.1.2	127.0.0.1	ENIP	42	Sourc
20	2.545946	192.168.1.2	127.0.0.1	ENIP	42	Sourc

Obrázok 7 – 9: Odchytenie skúmanej vzorky programom Wireshark

7.3 Overenie dát prostredníctvom web rozhrania nástroja SLAmeter

Obrázok 7 – 10 zobrazuje hodinový úsek z merania vykonaného nástrojom SLAmeter v reálnej sieťovej prevádzke na jednom zo serverov CNL laboratória Technickej univerzity v Košiciach. Interval pre agregáciu záznamov *aggregation trigger* A_t bol nastavený na hodnotu 250 milisekúnd a agregačná podmienka *aggregation condition* A_c bola nastavená na 5000 oktetov v toku.

Monitorovanie prebiehalo použitím toho istého monitorovacieho nástroja MyBeem s deaktivovaným a následne aj aktivovaným procesom agregácie. Dôvodom, prečo

meranie neprebiehalo na tom istom stroji simultálne dvoma exportérmi v tom istom čase, je možný výskyt interferencie medzi dvoma meracími procesmi. Keďže meranie bolo vykonávané na virtuálnych strojoch a oba exportéry používaju pre odchytávanie paketov *pcap* knižnicu, pri veľkých množstvách paketov dochádza k stratám paketov na oboch exportéroch, kedy systém nestíha obsluhovať tak veľké množstvo požiadaviek adresované od oboch exportérov. Zároveň kvôli obmedzeným technickým prostriedkom nebolo možné vykonať meranie na dvoch serveroch so zrkadlenou totožnou prevádzkou. To však nijako neovplyvňuje dôveryhodnosť výsledkov, alebo ich relevantnosť. Účelom pokusu bolo poukázať na rôzne hodnoty ukazovateľov v tom istom monitorovacom prostredí pri aktivovanom resp. deaktivovanom procese agregácie záznamov o tokoch a taktiež na zníženie záťaže ostatných prvkov monitorovacej architektúry.

V porovnaní s obrázkom 7 – 11 je potrebné si všimnúť najmä rozdielnu štruktúru grafu *Flow History*, ako aj počet zachytených tokov *Number of Flows* v priebehu tohto hodinového intervalu. Zároveň si môžme všimnúť, že hodnota prenesených dát *Transfered Data* a počet prenesených paketov *Transferred Packets*, ako aj ďalšie rýchlostné ukazovatele, niesú až na toľko rozdielne a teda je možné porovnanie týchto dvoch meraní považovať za relevantné. Tvar grafu *Flow History* pri zapnutom procese agregácie nieje natoľko hodnotovo hustý ako pri deaktivovanom procese. Je to zapríčinené tým , že v daných časových intervaloch nevzniká také veľké množstvo tokov, ale skôr sú udržiavané agregované záznamy o tokoch, u ktorých dochádza väčšinou k aktívnej expirácii. Hodnoty ukazovateľa *Number of Flows* sú pri aktivovanom procese agregácie tokov rádovo nižšie.

Z pokusu následne vyplíva, že podobné rozdiely by pretrvávali aj v dlhodobejšom časovom horizonte monitorovania. Pri upravovaní jednotlivých parametrov pre proces agregácie v konfiguračnom súbore je zároveň nutné dbať na to, že hodnoty daných parametrov od seba istým spôsobom závisia. Pretože ak nastavíme časový interval *aggregation trigger* A_t 5.4 na príliš nízku hodnotu a hodnota premennej hraničného počtu oktetov *aggregation condition* A_c 5.4 bude príliš vysoká, je možné, že merací

proces nezachytí v stanovenom časovom intervale dostatočné množstvo oktetov pre žiaden z tokov tak, aby tok nebol agregovaný. Preto bude efektívne dochádzať k agregácii každého odchyteného toku o jednu alebo viacero úrovní redukcie kľúčových hodnôt. Zároveň je potrebné dbať o to, aby hodnota intervalu *aggregation trigger* A_t bola menšia ako hodnota ukazovateľov pre pasívnu alebo aktívnu expiráciu toku.

7.4 Dopad výsledkov procesu agregácie na databázu

Výsledok procesu agregácie mal pozitívny dopad aj na databázu, v ktorej sú uložené záznamy o tokoch, čo sa prejavilo na nižšom počte uložených hodnôt ako aj výrazne rýchlejšej reakcii databázi na požiadavky analyzujúcej aplikácie. Nižšia hodnota ukazovateľa *Number of Flows* z obrázka 7 – 11 zároveň znamená, že v databáze sa nachádza značne menšie množstvo údajov pre časovo rovnaký monitorovaný interval.

Ako už bolo v tejto práci spomenuté, agregácia údajov má pozitívny dopad aj na

Tabuľka 7 – 2: Porovnanie počtu záznamov v databáze pri monitorovaní prevádzky v hodinovom časovom intervale

Proces agregácie (agregácia pod hodnotu)	Počet záznamov
deaktivovaná agregácia	23532 záznamov
aktivovaná agregácia (1000 oktetov/tok)	7091 záznamov
aktivovaná agregácia (5000 oktetov/tok)	4221 záznamov
aktivovaná agregácia (10000 oktetov/tok)	2735 záznamov

výkonnosť databázy, ktorá je v nástroji SLAmeter často krát kritickým miestom. Preto boli vykonané pokusy vyhodnotenia dotazov na databázu, pričom boli dotazované agregované a neagregované dáta. Jedná sa o dotazovanie dát z predošlých pokusných meraní trvajúcich jednu hodinu (použitá bola Mongo databáza, kde bol realizovaný výber spôsobom query - match - group), pričom každý pokus bol zopa-

kovaný 4 krát nad tou istou množinou dát.

Tabuľka 7 – 3: Čas dotazovania na agregované a neagregované dáta v databáze po hodinovom monitorovaní (čas v milisekundách)

Dotazovanie na dáta	Agregované dáta	Neagregované dáta
1.	2314 ms	2711 ms
2.	2233 ms	2700 ms
3.	2238 ms	2706 ms
4.	2219 ms	2753 ms

Taktiež bolo vykonané meranie trvajúce viac ako 24 hodín, pričom boli následne z databázy vyberané údaje (použitá bola Mongo databáza, kde bol realizovaný výber spôsobom query - match - group) presne z 24 hodinového časového úseku. Každý výber bol zopakovaný 4 krát. Výsledky pokusu sú znázornené tabuľkou 7 – 4.

Tabuľka 7 – 4: Čas dotazovania na agregované a neagregované dáta v databáze po 24 hodinovom monitorovaní (čas v milisekundách)

Dotazovanie na dáta	Agregované dáta	Neagregované dáta
1.	275368 ms	590854 ms
2.	272205 ms	585625 ms
3.	275149 ms	592641 ms
4.	273305 ms	583557 ms

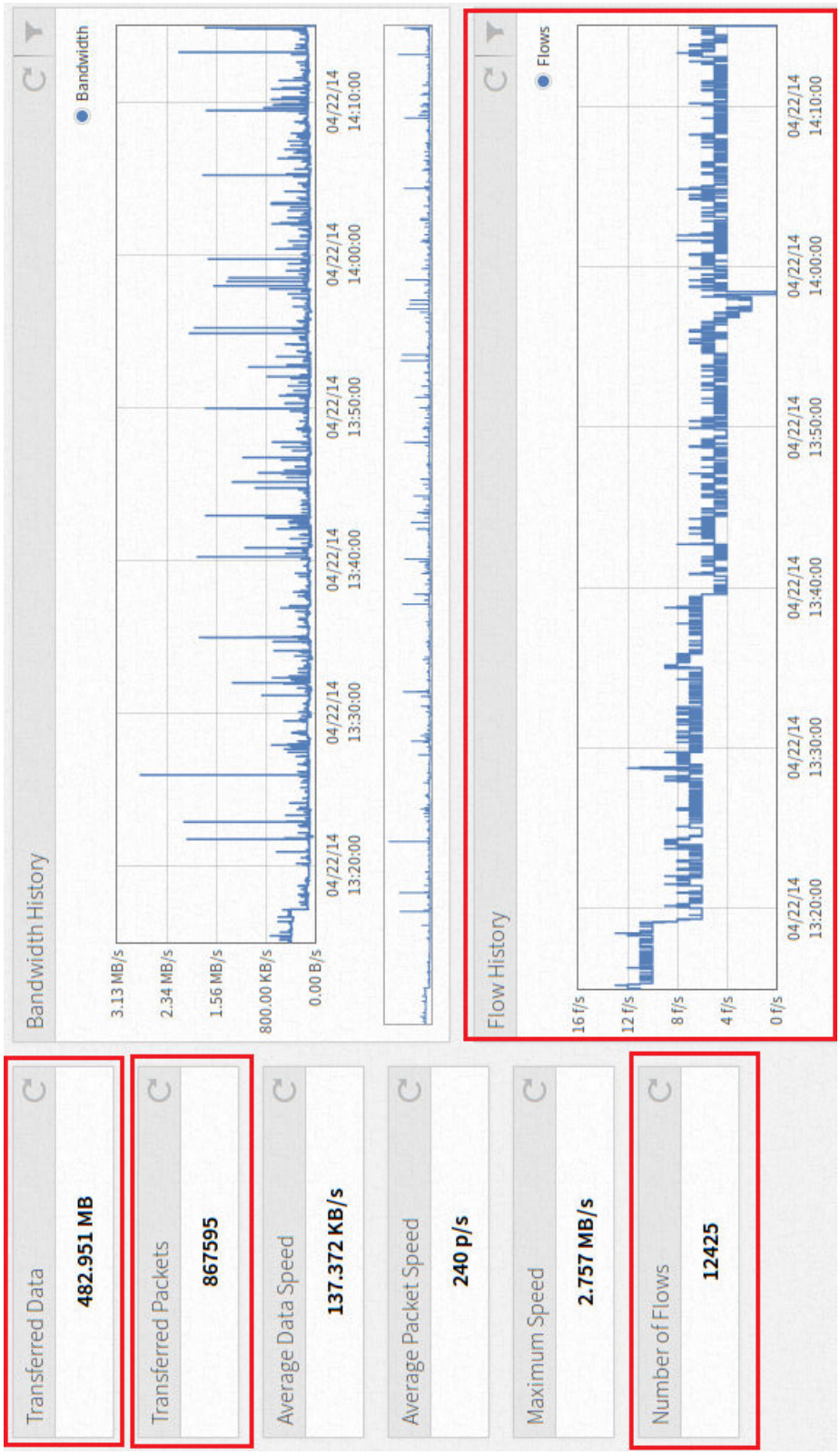
Tiež bol z databázy zistený údaj počtu uložených záznamov o tokoch z monitorovania trvajúceho 24 hodín (tab. 7 – 5).

Metóda rozpracovaná v tejto diplomovej práci predstavuje nový efektívny spôsob, ako pracovať s údajmi o tokoch v rozsiahlych počítačových sieťach s vysokým počtom tokov. V týchto sieťach totižto existuje mnoho miest, kde je možné monitorovať prevádzku v menej detailnom merítku. Takže pokiaľ prostredníctvom tejto metódy

Tabuľka 7 – 5: Porovnanie počtu záznamov v databáze pri monitorovaní prevádzky v 24 hodinovom časovom intervale

Proces agregácie (agregácia pod hodnotu)	Počet záznamov
deaktivovaná agregácia	4857729 záznamov
aktivovaná agregácia (10000 oktetov/tok)	438544 záznamov

dokážeme do istej miery kontrolovať množstvo tokov generovaných meracími bodmi (výsledky zobrazené v tab. 7 – 2 a tab. 7 – 5), následne skrátiť aj čas dotazovania sa na dáta v databáze (výsledky zobrazené v tab. 7 – 3 a tab. 7 – 4), je možné k jednému zhromažďovaciemu procesu v danom čase pripojiť väčšie množstvo meracích bodov nachádzajúcich sa v rôznych častiach siete a tým získať detailnejší obraz o situácii v jednotlivých častiach siete, ale aj o sieti ako celku v danom čase. Tento proces je možný, pretože pri danej rovnakej záťaži zhromažďovacieho procesu môžu záznami pochádzať z viacerých exportovacích procesov, keďže množstvo záznamov generovaných v exportéroch je do veľkej miery kontrolované agregačným mechanizmom.



Obrázok 7 – 10: Meranie nástrojom SLAmeter pri deaktivovanej agregácii tokov

Obrázok 7 – 11: Meranie nástrojom SLAmeter pri aktivovanej agregácii tokov

8 Záver

V tejto diplomovej práci bola podrobne analyzovaná, navrhnutá a napokon aj implementovaná a overená metóda redukcie informácii o IP tokoch v programe SLAmeter. V prvej kapitole, ktorou bola analýza danej problematiky, boli načrtnuté súčasné trendy v oblasti monitorovania počítačových sietí v spojitosti s redukciou informácií, pričom bolo poukázané na dôležitosť riešenia danej problematiky v dnešných konvergovaných sieťach. Zároveň bola daná problematika analyzovaná v spojitosti s nástrojom SLAmeter a nedostatkami, s ktorými sa bolo možné stretnúť pri monitorovaní počítačovej siete týmto nástrojom.

So zreteľom na analyzovanú problematiku bol navrhnutý celý mechanizmus agregácie tokov v nástroji SLAmeter ako aj úpravy nevyhnutné k správnemu fungovaniu celého procesu. Úpravy sa týkali najspodnejšej vrstvy nástroja SLAmeter, nástroja MyBeem.

Hlavným výsledkom tejto diplomovej práce je implementácia techniky na redukciu informácií o IP tokoch v programe MyBeem, čím sa výrazne zredukovala záťaž ostatných vrstiev architektúry a najmä databázy, čo prispelo k oveľa spoľahlivejšiemu fungovaniu celého nástroja a jeho vyššej nasaditeľnosti v rôznych počítačových sieťach. V programe MyBeem bolo potrebné implementovať úpravy týkajúce sa spôsobu zaraďovania paketov do tokov, modifikácie granularity zachytených tokov a taktiež modifikácie spravovania expirovaných tokov. Ďalej bolo potrebné navrhnúť a implementovať procesy, ktoré sa týkali definovania a stanovenia parametrov potrebných pre fungovanie procesu redukcie záznamov, redukcie kľúčových hodnôt toku, identifikácie rozsiahlych tokov, implementácie informačných elementov slúžiacich k skvalitneniu analýzy agregovaných tokov, ale najmä celkový proces agregácie záznamov o tokoch.

Ďalším významným výsledkom tejto práce je dosiahnutie vyššej konformity nástroja MyBeem so špecifikáciou IPFIX z pohľadu implementácie informačných elementov, ale najmä z pohľadu implementácie celého procesu agregácie záznamov o IP tokoch,

ktorá je aktuálne riešenou problematikou v oblasti monitorovania počítačových sietí. V záverečnej časti, ktorou bolo overenie, bola dokázaná správnosť implementovaného riešenia nasadením modifikovaného nástroja MyBeem do celkovej architektúry nástroja SLAmeter, ktorý bol následne použitý na monitorovanie reálnej sieťovej prevádzky a poukázaním na výstupy, ktoré tento nástroj ponúkol. Taktiež boli overenia realizované v parciálnych schémach, kde bol použitý nástroj MyBeem samostatne, ale aj v spojitosti so zhromažďovacím procesom a databázou nástroja SLAmeter.

V budúcnosti by bolo vhodné odtestovať metódu redukcie záznamov o IP tokoch v rôznych topológiách ako aj dátovo a rýchlostne rozmanitých prostrediach počítačových sietí s rôznym nastavením parametrov pre agregačný proces. V súvislosti s týmto testovaním bude potrebné odhaliť prípadné nedostatky, ktoré mohli vzniknúť testovaním implementácie na virtuálnych strojoch. Taktiež je potrebné odtestovať implementáciu riešenia tejto diplomovej práce v dlhodobom časovom horizonte a tak overiť jeho funkčnosť, prípadne zdokumentovať nedostatky, ktoré sa ukážu počas dlhodobého testovania.

V oblasti agregácie tokov je potrebné v budúcnosti implementovať redukcie informácií na základe časových úsekov a taktiež implementovať možnosť, aby exportér v procese agregácie nezasielal väčšie množstvo IPFIX správ, ako by bola stanovená kritická hodnota. Týmto by sa v podstate dosiahla agregácia tokov na základe exportovacej frekvencie a tak by bolo možné presnejšie stanoviť maximálny počet exportovacích procesov, ktoré je možné pripojiť k jednému zhromažďovaciemu procesu v danom čase. Táto práca síce vyriešila problém preplnenia vyrovnávacej pamäte tokov, avšak aj naďalej pretrváva pri nadmernej záťaži v programe MyBeem nedostatok v podobe prepĺňania vyrovnávacej pamäte odchytených paketov *packet cache*.

V neposlednom rade je potrebné podrobiť analýze aj konformitu nástroja MyBeem s ostatnými požiadavkami IPFIX protokolu a prípadné nájdené implementačné nedostatky upraviť.

Zoznam použitej literatúry

[1] CLAISE, B. -TRAMMELL, B. -AITKEN, P. 2004. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* RFC7011,2070-1721, ISSN, dostupné na internete: `http://tools.ietf.org/html/rfc7011`

[2] QUITTEK, J. -ZSEBY, T. -CLAISE, B. -ZEDNER, S. 2004. *Requirements for IP Flow Information Export (IPFIX)* RFC3917, dostupné na internete: `http://www.rfc-base.org/txt/rfc-3917.txt`

[3] SCHULZRINNE, H. -CASNER, S. -FREDERIC, R. -JACOBSON, V. 2003. *RTP: A Transport Protocol for Real-Time Applications* RFC3550, dostupné na internete: `http://www.ietf.org/rfc/rfc3550.txt`

[4] FELDMANN, A. et al. 2000. *Deriving traffic demands for operational IP networks: Methodology and experience.* In ACM SIGCOMM, Aug. 2000.

[5] TRAMMELL, B. -WAGNER, A. -CLAISE, B. 2013. *Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol* RFC7015, 2070-1721, ISSN, dostupné na internete: `http://tools.ietf.org/html/rfc7015`

[6] CLAISE, B. -CISCO SYSTEMS, INC. 2008. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information* RFC5101, dostupné na internete: `https://tools.ietf.org/html/rfc5101`

[7] IRINO, H. -KATAYAMA, M. -CHAKI, S. 2008. *A Study of Adaptive Aggregation on IPFIX* 978-4-88552-226-0,IEICE, 2008.

[8] HU, Y. -CHIU, Dah-M. -C.S. LUI, J. 2006. *Adaptive FLow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks* 1-4244-0143-7,IEEE, 2006.

[9] CHENG, G. -GONG, J. 2008. *Adaptive Aggregation Flow Measurement on high speed Links* 1-4244-2424-5,IEEE, 2008.

[10] ESTAN, C. -VARGHESE, G. 2002. *New Directions in Traffic Measurement and Accounting.* SIGCOMM'02, Pittsburgh, Pennsylvania, USA, August 19-23, 2002.

[11] DRESSLER, F. -GERHARD, M. 2006. *Flexible Flow Aggregation for Adaptive Network Monitoring* 1-4244-0419-3,IEEE, 2006.

[12] PEKÁR, A. 2013. *Modelovanie a návrh systémov pre monitorovanie sieťovej prevádzky.* Písomná práca k dizertačnej skúške, Košice: KPI FEI TUKE, 2013.

[13] KECSEY, T. 2012. *Optimalizácia meracieho a exportovacieho procesu nástroja BasicMeter* Písomná práca k diplomovej skúške, Košice: KPI FEI TUKE, 2012.

[14] TREMKO, S. 2012. *Meracie body pre nástroj SLA Meter.* Písomná práca k bakalárskej skúške, Košice: KPI FEI TUKE, 2012.

[15] MATHIS, M. -MAHDAVI, J. 1993. *Diagnosing internet congestion with a transport layer performance tool.* In *In Proceedings of INET'96*, 1996.

[16] PEKÁR, A. 2014. *Modeling and design of systems for network traffic monitoring.* Dizertačná práca, Košice: KPI FEI TUKE, 2014.

[17] CROVELLA, M. -KRISHNAMURTHY, B. 2006. *Internet Measurment: Infrastructure, Traffic and Applications.* John Wiley and Sons, Inc, 2006.

[18] TIRUMALA, A. -COTTRELL, L. -DUNIGAN, T. 1993. *Measuring end-to-end bandwidth with iperf using web100.* In *Web1000*,Proc. of Passive and Active Measurement Workshop, 2003.

[19] DOWNEY, A. 1999. *Using pathchar to estimate internet link characteristics.* In *In Proceedings of ACM SIGCOMM*, strany 241-250, 1990.

[20] STRAUSS, J. -KATABI, J. -KAASHOEK, F. 2003. *A measurement study of available bandwidth estimation tools.* In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, strany 39-44, New York, NY, USA, 2003. ACM.

[21] BOLOT, J.-C. 1993. *End-to-end packet delay and loss behavior in the internet.* In *Conference proceedings on Communications architectures, protocols and applications*,SIGCOM '93, strany 289-298, New York, NY, USA, 1993. ACM.

[22] BELLOVIN, S. M. 1992. *A best-case network performance model.* ATT Research, Tech. Rep,1992.

[23] MONICA 2013. *SLAmeter*, dostupné na internete: `http://wiki.cnl.sk/Monica/SLAmeter`

[24] SADASIVAN, G. -BROWNLEE, N. -CLAISE, B. -QUITTEK, J. 2009. *Architecture for IP Flow Information Export* RFC5470, dostupné na internete: `http://www.ietf.org/rfc/rfc5470.txt`

[25] KESHAV, S. 1991. *A control-theoretic approach to flow control.*,1991.

[26] FANG, W. -PETERSON, L. 1999. *Inter-as traffic patterns and their implications.*,In IEEE GLOBECOM, Dec. 1999.

[27] CISCO Systems 2003. *Sampled NetFlow* Cisco IOS Software Releases 12.0 S, dostupné na internete: `http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html`

[28] CISCO Systems 2008. *Cisco IOS Flexible NetFlow* Cisco IOS Flexible NetFlow Technology, dostupné na internete: `http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/ps6965/product_data_sheet0900aecd804b590b.html`

[29] CLAISE, B. -JOHNSON, A. -QUITTEK, J. -Cisco Systems, Inc. 2009. *Packet*

Sampling (PSAMP) Protocol Specifications RFC5476, dostupné na internete: `http://tools.ietf.org/html/rfc5476`

[30] ZSEBY, T. -MOLINA, M. -DUFFIELD, N. -NICCOLINI, S. 2009. *Sampling and Filtering Techniques for IP Packet Selection* RFC5475, dostupné na internete: `http://tools.ietf.org/html/rfc5475`

Zoznam príloh

Príloha A CD médium obsahujúce:

- diplomovú prácu v elektronickej podobe
- používateľskú a systémovú príručku v elektronickej podobe
- článok k diplomovej práci v elektronickej podobe
- zdrojové kódy nástroja MyBeem

Príloha B Používateľská príručka

Príloha C Systémová príručka

Príloha D Článok k diplomovej práci