Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky
Katedra počítačov a informatiky

Zhromažďovací proces nástroja BasicMeter

Bakalárska práca

Príloha A

SYSTÉMOVÁ PRÍRUČKA JXColl v3.5

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Vedúci práce:	Ing. Juraj Giertl, PhD.
Konzultant:	Ing. Martin Révés

Košice 2010 **Tomáš Vereščák**

Zoznam obrázkov

Zoznam tabuliek

1 Funkcia programu

Program JXColl (Java XML Collector) slúži na zachytávanie a predspracovávanie informácii o tokoch v sieťach získané exportérom. Je súčasťou meracej architektúry BasicMeter, ktorý na základe nastavených parametrov konfiguračného súboru vie dané údaje ukladať do databázy alebo ich sprístupniť pomocou vlastného protokolu pre priame spracovanie (protokol ACP) používateľovi. Údaje uložené v databáze slúžia pre neskoršie vyhodnotenie prídavnými modulmi spomínanej meracej architektúry a sú v súlade s požiadavkami protokolu IPFIX. JXColl tiež generuje účtovacie záznamy, ktoré slúžia na analýzu používania siete konkrétnym používateľom z hľadiska IP adries, protokolov, portov a časových charakteristík. Program bol vytvorený Ľubošom Koščom, neskôr zoptimalizovaný a doplnený novými funkciami Michalom Kaščákom, Adriánom Pekárom a Tomášom Vereščákom.

2 Analýza riešenia

Hlavným problémom v aplikácii JXColl v3 bolo abnormálne vyťaženie pamäte. Táto pamäť sa po určitom čase od spustenia aplikácie postupne vyčerpávala až narazila na svoje maximum. V tomto okamihu bola vyhodená chyba *java.lang.OutOfMemoryError : Java Heap Space*. Postupom, ktorý viedol k lokalizácii chyby bola analýza heap dump súboru, ktorý vygenerovala JVM pri páde JXColl. Pomocou Aplikácie Eclipse Memory Analyzer a jeho zobrazeniam *Dominator Tree* a *Path to GC roots* sme našli pôvod chyby. Je ňou trieda *IPFIXDataRecord* a zlé zaobchádzanie s jej inštanciou v triede *NetXMLParser*.

Druhým riešeným problémom bolo načítanie dát zo súboru ipfixFields.xml takým spôsobom, aby vlákna aplikácie, ktoré tieto dáta potrebujú, zbytočne neprehľadávali načítaný strom údajov (v reprezentácii Document Object Model), ale aby tieto údaje mohli byť prístupne s konštantnou časovou zložitosťou O(1).

Program JXColl bol doteraz ukončovaný násilne pomocou klávesovej kombinácie Ctrl+C. V tejto práci bol tento systém vypínania aplikácie zachovaný, avšak bolo implementované korektné ukončovanie vlákien, aby dáta ktoré už boli prijaté a uložené do PacketCache mohli byť ešte spracované. Zároveň týmto spôsobom sa pri ukončení najskôr odpojíme od databázy.

V programe boli taktiež nájdené chyby, ktoré umožňovali pád JXColl pri príjme informačných elementov, ktoré v súčasnosti nevie spracovať. Či už sa jedná o informačné elementy, ktoré sa nenachádzajú v súbore *ipfixFields.xml*, a teda nie je možné dáta dekódovať, alebo sa jedná o dáta, o ktorých informácie síce máme, ale sú v rozpore z reálne prijatými dátami. Takáto situácia môže nastať v prípade ak informačný element prijatý pomocou JXColl je iného dátového typu ako je uvedené v súbore *ipfixFields.xml*. Môže nastať aj situácia, že informačný element je podporovaný v *ipfixFields.xml*, avšak dátový typ tohto informačného elementu JXColl nevie dekódovať. Vo všetkých týchto prípadoch JXColl reagoval pádom, prípadne preskočením zvyšných informačných elementov IPFIX dátového záznamu, ktoré mohli byť v poriadku. Je potrebná náprava tejto skutočnosti.

3 Popis programu

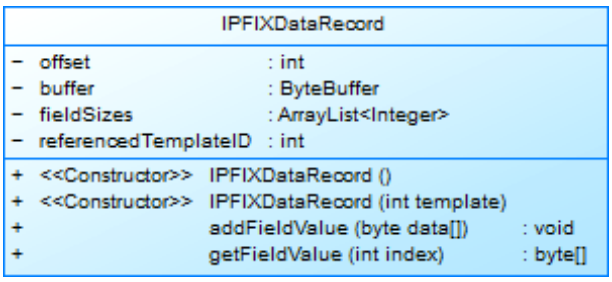
Jednotlivé časti programu sú umiestnené v nasledujúcich balíkoch:

- sk.tuke.cnl.bm.JXColl.export - triedy určené na export dát do databázy alebo protokolom ACP
- sk.tuke.cnl.bm.JXColl.IPFIX - triedy s manuálnou implementáciou IPFIX
- sk.tuke.cnl.bm.JXColl - triedy samotného programu
- sk.tuke.cnl.bm - spoločné triedy pre celý projekt

3.1 Popis riešenia

Problém s vyčerpaním pamäte bol spôsobený nesprávnym používaním objektu *IPFIXDataRecord*, v porušení objektového princípu. Objekty by mali byť zapuzdrené, to znamená, že dátové členy triedy by nemali byť priamo prístupné okoliu, ale len cez ich rozhranie, ktorými sú metódy. V tomto prípade bol tento princíp porušený a to nakoniec viedlo k vzniku memory leak-u. Inštancia triedy *IPFIXDataRecord* bola vytvorená v triede *NetXMLParser*, kde sa uložila do členskej premennej *ipfixData*. Tento člen označuje aktuálnu pozíciu „kurozra“ v internej štruktúre IPFIX dátového záznamu (buffer). Pri pridaní informačného elementu pomocou metódy addFieldValue(byte[] data) sú predané dáta vložené do buffra na aktuálnu pozíciu označenú kurozorom (offset). Zároveň je o veľkosť týchto dát offset inkrementovaný a do zoznamu fieldSizes typu *java.util.ArrayList<Integer>* je vložená číselná hodnota veľkosti aktuálne pridaných dát, aby uložené dáta mohli byť zrekonštruovateľné. Keďže členy triedy *IPFIXDataRecord* *offset*, *buffer* aj *fieldSizes* boli verejne prístupné, bol v mylnej domnienke správnosti tohto riešenia objekt znovupoužívaný vynulovaním člena *offset*. Tým sa zabezpečilo, že nové dáta síce znovu budú ukladané od začiatku, avšak k vyprázdneniu obsahu zoznamu fieldSizes nedochádzalo. Dáta ukladané do tejto kolekcie spôsobovali zahltenie pamäte a preto musel byť spôsob použitia tohto objektu zmenený. Riešenie problému pozostávalo v nastavení súkromnej viditeľnosti členov offset, buffer a fieldSizes triedy *IPFIXDataRecord* a v triede *NetXMLParser* vytváraním nového objektu *IPFIXDataRecord* pri získaní každého dátového záznamu z prijatej IPFIX správy. Upravná trieda *IPFIXDataRecord* je zobrazená na obrázku 3 – 1 Základné triedy spracúvajúce IPFIX správy ukladané do PacketCache sú uvedené na obrázku 3 – 2

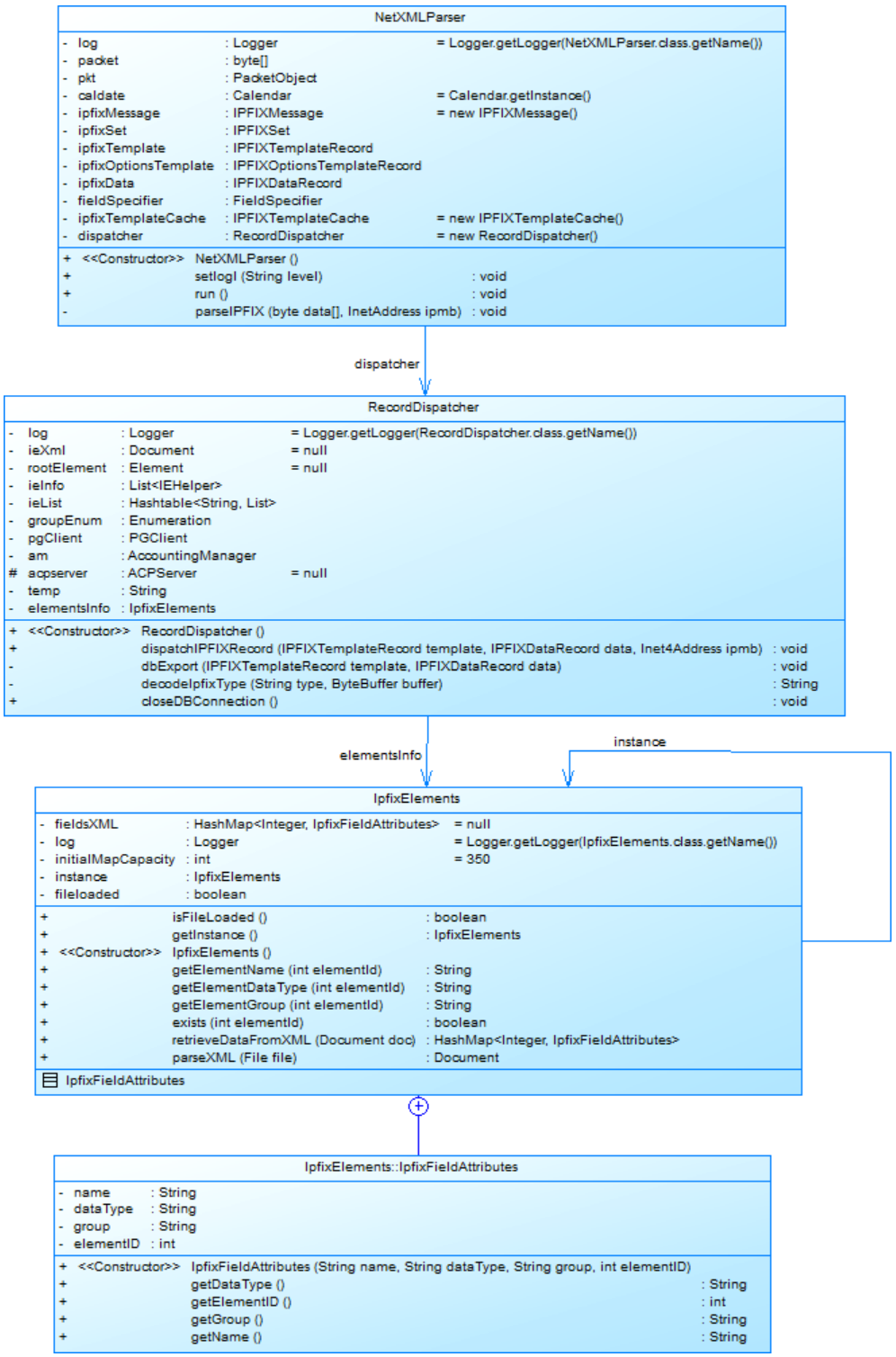
Na zjednodušenie prístupu k informačným elementom v aplikácii bola vytvorená trieda *IpfixElements*. Tá pri vytvorení svojej inštancie načíta údaje z XML súboru *ipfixFields.xml* pomocou metódy *parseXML(File file)*. Argumentom tejto funkcie je súbor ktorý chceme spracovať a cesta k nemu je nastavená v konfiguračnom súbore



Obrázok 3 – 1 Trieda IPFIXDataRecord

jxcoll.conf v položke XMLFile. Ak sa tento súbor nenájde, automaticky sa predpokladá cesta */etc/jxcoll/ipfixFields.xml*. V pamäti sa tak vytvorí strom objektov reprezentujúci jednotlivé elementy XML súboru. Pomocou metódy *retrieveDataFromXML(org.w3d.dom.Document doc)* tieto dáta vložíme v požadovanom tvare do kolekcie typu *java.util.HashMap*. Táto kolekcia umožňuje vytvoriť páry kľúč, hodnota. Kľúčom v tomto prípade je ID informačného elementu (Integer), tak ako je definované v súbore ipfixFields.xml. Hodnotou je objekt typu IpfixFieldAttributes, ktorý predstavuje opis jedného informačného elementu. Objekt typu IpfixFieldAttributes obsahuje informácie o názve informačného elementu, jeho IPFIX dátovom type, skupine elementu a jeho ID. Na prístup k dátam slúžia metódy *getElementName(int elementID)*, *getElementDataType(int elementID)*, *getElementGroup(int elementID)*. Na zistenie, či daný informačný element je podporovaný slúži metóda *exists(int elementID)*. Kolekcia Hashmap je nesynchronizovaná, to môže spôsobiť problémy v prípade ak by viacero vlákien do nej zapisovalo údaje. Avšak z princípu spracovania tohto XML súboru v JXColl je zrejmé, že k načítaniu údajov dôjde len pri spustení aplikácie, ďalšie prístupy ku kolekcii sú len na operáciu čítania. Skutočnosť, že kolekcia je nesynchronnizovaná znamená, že prístup do nej je rýchlejší ako pri synchronizovaných kolekciách. Inštancia triedy *IpfixElements* môže byť v aplikácii iba jedna, čo je zabezpečené návrhovým vzorom Singleton.

Korektné ukončovanie je implementované systémom prerušení (angl. interrupts). Na ukončenie vlákna je potrebné zavolať nad ním metódu interrupt(). Tá nastaví stav prerušenia (angl. interrupt status) a v hlavnom cykle vlákna je potrebné zisťovať



Obrázok 3 – 2 Základné triedy spracúvajúce dáta uložené v PacketCache

stav tejto premennej. Pri jej nastavení sa má vlákno ukončiť. V prípade ak je v tele hlavného cyklu vlákna volaná blokujúca operácia, pri prerušení táto vyhodí výnimku InterruptedException. Túto je potrebné odchytiť a znovu prerušiť vlákno, pretože pri vyhodení výnimky *InterruptedException* sa stav prerušenia vynuluje. Tým docielime neplatnosť podmienky hlavného cyklu vlákna a to sa korektne ukončí. V prípade JXColl sa najskôr ukončuje vlákno *PacketListener*, ktoré slúži na prijímanie paketov zo siete a ukladanie IPFIX paketov do PacketCache. Ukončíme ho zavolaním metódy *interrupt()* nad objektom vlákna. Potom čakáme na ukončenie vlákna pomocou volania *join()*. Následne čakáme na vyprázdnenie PacketCache a potom prerušíme ostatné spustené vlákna.

Nekorektné vstupné dáta môžu ovplyvniť stabilitu JXColl, do takej miery, že program môže abnormálne ukončiť svoju činnosť. Na zabránenie týmto stavom pri dekódovaní dátového typu informačného elementu, v prípade ak na vstupe je element, ktorý nie je podporovaný (nie je uvedený v ipfixFields.xml), metóda spracúvajúca dátové záznamy (exportData) triedy RecordDispatcher vypíše chybovú správu a informačný element sa preskočí. Pri odchytení sa vypíše chybová správa a informačný element sa preskočí. Podobne, ak dátový typ nie je možné dekódovať v metóde *decodeIPFIXData* (IPFIX dátový typ nie je implementovaný), vyhodí sa výnimka *UnsupportedDataTypeException* a po jej odchytení v metóde exportData je zobrazená chybová správa a informačný element sa preskočí. V prípade, ak veľkosť vstupných dát nekorešponduje s očakávaným dátovým typom, je výhodená výnimka *BufferUnderFlowException*. Tá sa odchytí v metóde *exportData*, vypíše sa chybová správa a informačný element sa preskočí. Takto sa JXColl zotaví z týchto neočakávaných dát.

3.2 Popis tried, členských premenných a metód

Kedže sa počas vývoja a odstraňovania chýb pôvodné triedy a ich metódy nezmenili, v nasledujúcich častiach budú uvedené len tie, ktoré boli doplnené počas riešenia jednotlivých problémov uvedených v analýze riešenia. Popis ostatných tried a metód je uvedený v príručkach predošlých verzií programu.

3.2.1 Trieda IpfixElements

Trieda poskytuje funkcie na pohodlné získavanie potrebných údajov o IPFIX informačných elementoch. Trieda načítava XML dokument ipfixFields.xml do pamäte a vloží údaje o jednotlivých IPFIX elementoch do kolekcie typu HashMap s mapovaním ID informačného elementu na objekt typu IpfixFieldAttributes, ktorý obsahuje konkrétne informácie o danom elemente - meno, dátovy typ, skupinu, ID.

Konštruktor

public ***IpfixElements****()*

Získa inštanciu tejto triedy, povolená je len jedna inštancia (Singleton).

Metódy

public static boolean ***isFileLoaded****()*

Slúži pre iné vlákna na zistenie, či načítanie súboru prebehlo bez chýb.

Návratová hodnota:

true ak načítanie dát prebehne v poriadku, inak false

public HashMap<Integer, IpfixFieldAttributes> ***retrieveDataFromXML****(Document doc)*

Údaje sparsované pomocou DOM-u vloží do kolekcie typu HashMap kvôli rýchlejšiemu prístupu k jednotlivým atribútom. Kľúčom kolekcie je číslo - identifikátor IPFIX informačného elementu, hodnotou kolekcie je objekt typu IpfixFieldAttributes. Ten obsahuje len informácie využívané v JXColl - name, group, datatype, elementId.

Parametre:

doc - DOM dokument (stromová štruktúra XML elementov)

Návratová hodnota:

HashMap<Integer, IpfixFieldAttributes> - naplnená kolekcia obsahujúca informácie o informačných elementoch

public org.w3c.dom.Document ***parseXML****(File file)* throws org.xml.sax.SAXException, java.io.IOException, javax.xml.parsers.ParserConfigurationException

Sparsuje dodany xml súbor a načíta ho do pamäte.

Parametre:

file - súbor ktorý chceme načítať do pamäte

Návratová hodnota:

org.w3c.DOM.Document - in-memory reprezentácia XML dokumentu

Výnimky:

org.xml.sax.SAXException - v prípade chyby počas parsovania (nesprávny formát XML súboru)
java.io.IOException - v prípade chyby počas čítania súboru

avax.xml.parsers.ParserConfigurationException - ak nemohol byť vytvorený objekt DocumentBuilder slúžiaci na reprezentáciu údajov

public String ***getElementName****(int elementId)*

Získa názov informačného elementu podľa zadaného jeho ID

Parametre:

elementID - číslo informačného elementu

Návratová hodnota:

String - názov informačného elementu

public String ***getElementDataType****(int elementId)*

Získa IPFIX dátový typ informačného elementu podľa zadaného jeho ID

Parametre:

elementID - číslo informačného elementu

Návratová hodnota:

String - IPFIX dátový typ informačného elementu

public String ***getElementGroup****(int elementId)*

Získa skupinu informačného elementu podľa zadaného jeho ID

Parametre:

elementID - číslo informačného elementu

Návratová hodnota:

String - skupina informačného elementu

*public boolean **exists**(int elementId)*

Parametre:

elementID - číslo informačného elementu

Zistí, či v XML súbore existuje položka s daným ID

Návratová hodnota:

true, ak informačný element existuje, inak false

3.2.2 Trieda IpfixFieldAttributes

Pomocná trieda, ktorá uchováva informácie o informačnom elemente: meno, dátovy typ, skupinu a číslo IPFIX informačného elementu.

Konštruktor

*public **IpfixFieldAttributes**(String name, String dataType, String group, int elementID)*

Vytvorí objekt reprezentujúci jeden informačný element.

Parametre:

name - názov informačného elementu
dataType - dátový typ informačného elementu
group - skupina informačného elementu
elementID - ID informačného elementu

Metódy

public String ***getName****()*

Získa názov informačného elementu

Návratová hodnota:

String - názov informačného elementu

public String ***getDataType****()*

Získa dátový typ informačného elementu

Návratová hodnota:

String - dátový typ informačného elementu

public String ***getGroup****()*

Získa skupinu informačného elementu

Návratová hodnota:

String - skupina informačného elementu

public int ***getID****()*

Získa ID informačného elementu

Návratová hodnota:

int - ID informačného elementu

4 Preklad programu

4.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe bakalárskej práce.

Sú k dispozícii tieto zdrojové texty:

- balíček sk.tuke.cnl.bm:
  - Filter.java
  - InetAddr.java
  - InvalidFilterRuleException.java
  - SimpleFilter.java
  - ACPIPFIXTemplate.java
  - Templates.java
- balíček sk.tuke.cnl.bm.JXColl:
  - Config.java
  - IJXConstants.java
  - IpfixElements.java
  - JXColl.java
  - NetConnect.java
  - NetXMLParser.java
  - OutputCache.java
  - PacketCache.java
  - PacketListener.java
  - PacketObject.java
  - RecordDispatcher.java
  - SaxXSParser.java
  - XMLNode.java
  - XMLPacket.java
- balíček sk.tuke.cnl.bm.JXColl.export:
  - ACPServer.java
  - ACPIPFIXWorker.java
  - DBExport.java
  - PGClient.java

- balíček sk.tuke.cnl.bm.JXColl.accounting:
  - AccountingManager.java
  - AccountingRecord.java
  - AccountingRecordsCache.java
  - AccountingRecordsExporter.java

- balíček sk.tuke.cnl.bm.JXColl.IPFIX:
  - FieldSpecifier.java
  - IPFIXDataRecord.java
  - IPFIXMessage.java
  - IPFIXOptionsTemplateRecord.java
  - IPFIXSet.java
  - IPFIXTemplateRecord.java
  - IPFIXTemplateCache.java

4.2 Požiadavky na technické prostriedky pri preklade

Preklad programu si vyžaduje nasledovnú hardvérovú konfiguráciu:

- CPU Intel Pentium III 1Ghz alebo ekvivalent
- operačná pamäť 512MB
- pevný disk s 100MB voľného miesta
- grafická karta novej generácie s minimálne 64MB pamäťou

4.3 Požiadavky na programové prostriedky pri preklade

- ľubovoľný operačný systém s podporou Java Virtual Machine (JVM) (Windows XP/Server 2003/Vista, GNU/Linux alebo Solaris)
- Java Runtime Environment (JRE) verzie 1.6.0 a vyššej
- knižnice dodávané na inštalačnom médiu

4.4 Náväznosť na iné programové produkty

Program umožňuje ukladanie dát do databázy alebo ich sprístupnenie priamym pripojením, ktoré budú následne vyhodnotené príslušnými prídavnými modulmi. Je

implementáciou zhromažďovacieho procesu architektúry BasicMeter. Z toho vyplýva jeho náväznosť na merací a exportovací proces - BEEM.

4.5 Vlastný preklad

Preklad programu spočíva v nakopírovaní zdrojových súborov a spustení kompilátora jazyka Java s potrebnými parametrami a parametrom classpath nastaveným na prídavné knižnice. Odporúča sa použiť váš obľubený java IDE, kde stačí jednoducho nastaviť verziu JDK na 6.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam.

4.6 Opis známych chýb

V súčasnosťi nie sú známe žiadne vážne chyby.

5 Zhodnotenie riešenia

Hlavným cieľom práce bolo zvýšiť spoľahlivosť zhromažďovacieho procesu nástroja BasicMeter. Základným problémom brzdiacim vývoj a nasadenie nástroja BasicMeter bola chyba spojená s vyčerpaním voľnej pamäte programu. Náprava uvedená v popise riešenia sa ukázala ako správna, no JXColl je aplikácia určená na nepretržitý beh a nie je možné vylúčiť, že v programe sa skrýva nejaká iná chyba. Doposiaľ namerané výsledky to však nepotvrdzujú. Zotavenie z neočakávaných stavov napomohlo k stabilite zhromažďovacieho procesu a vďaka korektnému ukončovaniu aplikácie sa nespracované IPFIX pakety z PacketCache v čase požiadavky na vynutie stihnú pred ukončením programu spracovať. Zároveň zjednodušený prístup k informačným elementom predstavuje zvýšenie efektivity spracovania dátových záznamov. Tento je vhodné v budúcnosti implementovať aj do časti ACP pripojenia. V snahe zamerať sa na lokalizáciu pamäťového problému bola v minulosti odstránená podpora protokolu NetFlow. Pre budúci vývoj BasicMetra je vhodné v budúcnosti doplniť podporu tohto protokolu.

6 Zoznam použitej literatúry

[1]Koščo, M.: Opis sieťových protokolov prostredníctvom jazyka XML, 2005, Diplomová práca, KPI FEI TU, Košice

[2]Kaščák, M.: Príspevok k problematike aplikačného využitia meraní prevádzkových parametrov počítačových sietí, 2007, Diplomová práce, KPI FEI TU, Košice

[2]Pekár, A.: Meranie prevádzkových parametrov site v reálnom čase, 2009, Bakalárska práca, KPI FEI TU, Košice