

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

**Aplikačný rámec pre sprostredkovanie IPFIX
správ v nástroji SLAmeter**

Diplomová práca

Príloha A

SYSTÉMOVÁ PRÍRUČKA Mediator v1.0

Študijný program: Informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Vedúci práce: Ing. Peter Fecilak, PhD.
Konzultant: Ing. Adrián Pekár

Košice 2013

Bc. Rastislav Kudla

Copyright © 2013 Rastislav Kudla. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Text. A copy of the license can be found at <http://www.gnu.org/licenses/fdl.html>.

Obsah

1	Funkcia programu	6
2	Analýza problému	6
2.1	Vybrané príklady použitia sprostredkovania správ	7
2.2	Analýza aplikačného rámca pre IPFIX Mediátor	8
3	Popis programu	10
3.1	Popis riešenia	10
4	Popis tried, členských premenných a metód	10
4.1	Balík sk.tuke.cnl.bm.Mediator	11
4.1.1	Trieda Default	11
4.2	Balik sk.tuke.cnl.bm.Mediator.collecting	11
4.2.1	Trieda UDPServer	11
4.2.2	Trieda IpfixParser	12
4.2.3	Trieda IpfixDecoder	12
5	Preklad programu	19
5.1	Zoznam zdrojových textov	19
5.2	Požiadavky na technické prostriedky pri preklade	20
5.3	Požiadavky na programové prostriedky pri preklade	20
5.4	Náväznosť na iné programové produkty	21
5.5	Vlastný preklad	21
5.6	Vytvorenie inštalačného DEB súboru	21
5.7	Opis známych chýb	23
6	Zhodnotenie riešenia	24
	Zoznam použitej literatúry	24

Zoznam obrázkov

2–1 Referenčný model sprostredkovania správ v IPFIX	8
2–2 Zjednodušený model komponentov IPFIX Mediátora	9

Zoznam tabuliek

1 Funkcia programu

Program Mediator je implementáciou aplikacneho ramca pre problem sprostredkovania sprav v protokole IPFIX (*IP Flow Information Export (IPFIX) Mediation Problem*) vyvijany vyskumnou skupinou MONICA sidliacou v Laboratoriu pocitacovych sieti (CNL) na Technickej Univerzite v Kosiciach. Je sucastou meracej architektury SLAmeter, ktorej ulohou je pasivne meranie parametrov sietovej prevadzky na baze tokov. Na zaklade nameranych hodnot urcuje triedu kvality sluzieb a Internetoveho pripojenia poskytovatelov Internetu. Trieda kvality vypoveda o dodrziavani zmluvy o urovni poskytovanej sluzby - *SLA*.

Mediaotrebnú pre široký rad meracích aplikácii. Sprostredkovateľske moduly Mediatora mozu z pohľadu manipulácie s dátami poskytovať agregáciu, koreláciu, filtrovanie, anonymizáciu a iné úpravy záznamov o tokoch za účelom šetrenia výpočtových zdrojov meracieho systému a vykonávania predspracovania úloh pre kolektor. Z hľadiska interoperability nástrojov rôznych vývojárov, môžu poskytovať konverziu z iných protokolov na IPFIX, respektíve zvyšovať spoľahlivosť exportov napríklad prevodom z nespoľahlivého, bezspojoovo orientovaného protokolu UDP na spoľahlivý SCTP.

Program bol v roku 2013 vytvorený Rastislavom Kudlom v rámci jeho diplomovej práce.

2 Analýza problému

Problematika sprostredkovania IPFIX správ je podrobne spracovaná v [1]. Hovorí o tom, že sieťoví administrátori často celia problémom týkajúcim sa škálovateľnosti meracieho systému, flexibility monitorovania na základe tokov, alebo aj spoľahlivosti exportovania. Napriek tomu, že sa vyvinuli známe techniky ako *uzorkovanie a filtrovanie paketov*, *zoskupovanie dátových záznamov*, alebo *replikácia exportu*, tieto

problémy nevymizli. Pozostávajú z prispôsobovania niektorých parametrov meracích nástrojov zdrojom meracieho systému zatiaľ čo musia naplniť patričné podmienky ako sú *presnosť nameraných dát*, *granularita toku*, či *spolahlivosť exportu*. Tieto okolnosti závisia na dvoch faktoroch:

1. **Kapacita meracieho systému** - pozostáva zo šírky pásma spravovanej siete, kapacity úložiska a výkonu exportovacích a zhromažďovacích nástrojov
2. **Požiadavky aplikácie** - rôzne aplikácie vyžadujú rôznu zrnitosť záznamov o tokoch a presnosť dát.

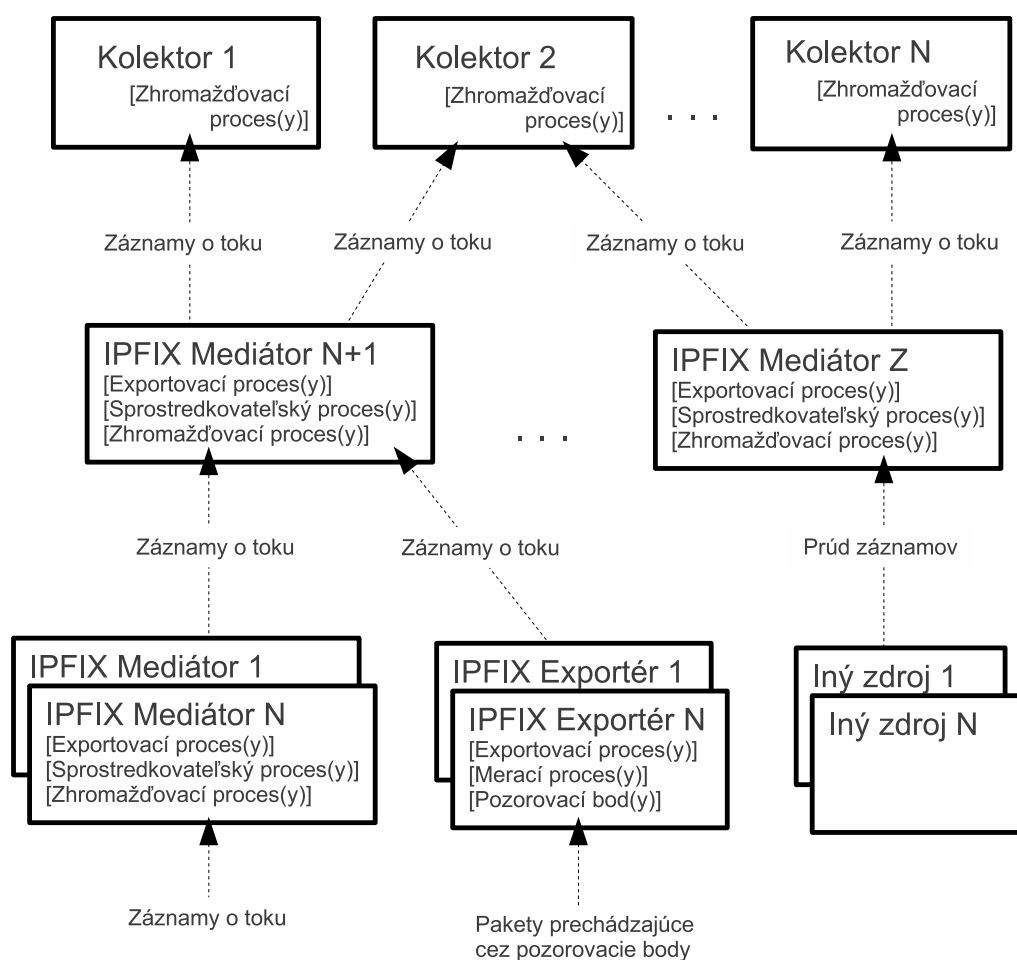
2.1 Vybrané príklady použitia sprostredkovania správ

RFC 5982 [1] uvádza viacero príkladov zaradenia IPFIX Mediátora do klasickej exportér - kolektor architektúry. Uvedme aspoň niektoré:

- prispôsobovanie granularity tokov,
- distribuovaná zhromažďovacia infraštruktúra,
- spájanie času,
- spájanie priestoru,
 - spájanie priestoru v rámci jednej pozorovacej domény,
 - spájanie priestoru viacerých pozorovacích domén jedného exportéra,
 - spájanie priestoru niekoľkých exportérov,
 - spájanie priestoru administratívnych domén,
- anonymizácia dátových záznamov,
- distribúcia dátových záznamov,
- konverzia z protokolu nižšej verzie na IPFIX,

2.2 Analýza aplikačného rámca pre IPFIX Mediátor

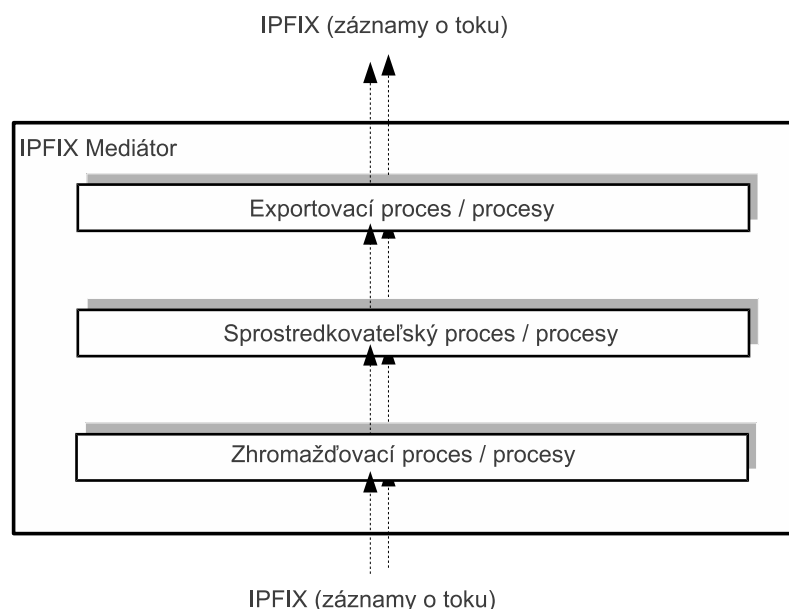
Analýze aplikačného rámca pre sprostredkovanie správ v IPFIX sa venuje RFC 6183 [2]. Na Obrázku 2–1 je zobrazený referenčný model sprostredkovania správ v IPFIX ako rozšírenie referenčného modelu IPFIX, popísaného v *Architecture for IP Flow Information Export* [3]. Táto schéma zobrazuje možné scenáre, ktoré môžu existovať v meracej architektúre.



Obr. 2–1 Referenčný model sprostredkovania správ v IPFIX

Funkčné komponenty v rámci každej entity sú ohraničené zátvorkami []. Mediátor môže prijímať záznamy o toku od iných mediátorov a exportérov a prúd záznamov z iných zdrojov. Za iné zdroje sa považujú nástroje iných protokolov, ako napríklad

NetFlow exportéry [4]. Spracované dáta vo forme záznamov o toku potom exportuje jednému alebo viacerým kolektorom a mediátorom.



Obr. 2–2 Zjednodušený model komponentov IPFIX Mediátora

Zjednodušený model komponentov IPFIX mediátora je predstavuje Obrázok 2–2. Mediátor obsahuje jeden alebo viac sprostredkovateľských procesov, hierarchicky uložených medzi jedným alebo viacerými exportovacími a zhromažďovacími procesmi. Tento model sa týka najbežnejšieho prípadu, kedy mediátor prijíma dátové záznamy od exportéra, alebo iného mediátora.

Sprostredkovateľské procesy sú kľúčovými funkčnými blokmi sprostredkovania správ v IPFIX. Musia pokryť každý príklad použitia sprostredkovania správ z kapitoly 2.1. Mediátor je schopný súčasne podporovať viac ako jeden sprostredkovateľský proces.

- **Paralelné spracovanie** - Prúd záznamov je spracovaný viacerými sprostredkovateľskými procesmi paralelne. V tomto scenári, každý sprostredkovateľský proces dostáva kópiu celého prúdu záznamov ako vstup.
- **Sériové spracovanie** - Sprostredkovateľské procesy sú zapojené sériovo. Výstupný prúd záznamov jedného procesu je vstupným prúdom nasledujúceho.

3 Popis programu

Jednotlivé časti programu sú umiestnené v nasledujúcich balíkoch:

- `sk.tuke.cnl.bm.Mediator.collecting` - implementacia zhromazdovacieho procesu
- `sk.tuke.cnl.bm.Mediator.exporting` - implementacia exportovacieho procesu
- `sk.tuke.cnl.bm.Mediator.IntermediateProcesses` - triedy tvoriace podporu pre sprostredkovateľské procesy. Hlavné triedy nových modulov musia byť umiestnené v tomto balíku!
- `sk.tuke.cnl.bm.Mediator.IPFIX` - triedy s manuálnou implementáciou protokolu IPFIX
- `sk.tuke.cnl.bm.Mediator.exceptions` - vlastné výnimky aplikácie
- `sk.tuke.cnl.bm.Mediator` - hlavné triedy samotného programu

3.1 Popis riešenia

4 Popis tried, členských premenných a metód

Kedže niektoré triedy Mediatora sú kvôli jednotnosti riešenia v rámci výskumnej skupiny MONICA totožné s triedami nástroja JXColl, v nasledujúcich častiach budú uvedené len tie, ktoré sa týkajú výhradne Mediatora. Popis ostatných tried a metód je uvedený v systemovej príručke programu JXColl [5].

4.1 Balík `sk.tuke.cnl.bm.Mediator`

4.1.1 Trieda `Default`

Trieda predstavuje rozhranie obsahujúce východiskové hodnoty konfiguračného súboru. Neobsahuje konštruktor ani žiadne metódy, iba verejne prístupné statické konštanty.

4.2 Balík `sk.tuke.cnl.bm.Mediator.collecting`

4.2.1 Trieda `UDPServer`

Slúži ako UDP server. Prijíma UDP datagramy cez `DatagramChannel` a uklada ich do `PacketCache`.

Konštruktor

*public **UDPServer**(int port)*

Konštruktor inicializuje `DatagramChannel`, nastaví mu blokový režim a priviaže ho k portu definovanom v konfiguračnom súbore, ktorý mu je predaný ako parameter. Nastaví meno vlákna.

Parametre:

port - číslo portu

Metódy

*public void **run**()*

Hlavná metóda vlákna. Pokiaľ nedôjde k prerušeniu, prijíma cez vytvorený kanál data od exportéra. Prijaté data obali do objektu `ByteBuffer` a predá ich spolu s časom prijatia a IP adresou a portom exportera metóde `write()`, ktorá ich zapíše do `PacketCache`.

*public void **cleanUp**()*

Táto metóda zruší čistiace vlákno pre UDP Template Cache. Je volaná pri prerušení tohto vlákna.

4.2.2 Trieda IpfixParser

Táto trieda sa používa na parsovanie IPFIX správ a ich spracovanie. V porovnaní s verziou v aplikácii JXColl bola precistená. Boli vypustené sekcie spracovávajúce TCP a SCTP spojenia. Zasadnejšia zmena prišla na výstupe z triedy. Sparsované dátové záznamy sú zabalené do vytvoreného objektu triedy **IPFIXFlowRecord**, spolu s príslušnou šablónou a hlavičkou prijatej IPFIX spravy. Vytvorený záznam o toku je spolu s reťazcom predstavujúcim zdroj záznamu (v tomto prípade „exporter“) posunutý triede **FlowRecordDispatcher**.

4.2.3 Trieda IpfixDecoder

Trieda so statickými metódami slúžiacimi na dekodovanie dát z dátového záznamu.

Metódy

*public String **decode**(String type, ByteBuffer buffer)*

Dekóduje dátový typ obsiahnutý v buffri do podoby reťazca podľa špecifikácie IP-FIX. Priamo nevykonáva dekodovanie, volá konkrétne metódy podľa kategórie dátového typu.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.

buffer - samotné dáta, ktoré sú predmetom dekodovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

UnsupportedDataException - Ak dátový typ nie je podporovaný

*public String **decodeUnsignedIntegralType**(String type, ByteBuffer buffer)*

Dekóduje celočíselné bezznamiekové dátové typy unsigned8, unsigned16, unsigned32 a unsigned64. Berie ohľad na skrátené dátové typy.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.

buffer - samotné dáta, ktoré sú predmetom dekodovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

`DataException` - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

*public String **decodeSignedIntegralType**(String type, ByteBuffer buffer)*

Dekóduje celočíselné znamienkové dátové typy `signed8`, `signed16`, `signed32` a `signed64`. Berie ohľad na skrátené dátové typy.

Parametre:

`type` - reťazec definujúci dátový typ obsiahnutý v buffri.

`buffer` - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

`DataException` - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

*public String **decodeFloatType**(String type, ByteBuffer buffer)*

Dekóduje desatinné dátové typy `float32` a `float64`. Berie ohľad na skrátené dátové typy.

Parametre:

`type` - reťazec definujúci dátový typ obsiahnutý v buffri.

`buffer` - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

UnsupportedDataException - Ak dátový typ nie je podporovaný

```
public String decodeAddressType(String type, ByteBuffer buffer)
```

Dekóduje dátové typy obsahujúce adresy: ipv4Address, ipv6Address a macAddress.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.

buffer - samotné dáta, ktoré sú predmetom dekodovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

UnsupportedDataException - Ak dátový typ nie je podporovaný

*public String **decodeBooleanType**(ByteBuffer buffer)*

Dekóduje boolean reprezentujúci pravdivostnú hodnotu.

Návratová hodnota:

Reťazec reprezentujúci pravdivostnú hodnotu, "true" alebo "false".

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ, alebo ak obsahuje inú hodnotu ako 0 alebo 1

UnsupportedDataException - Ak dátový typ nie je podporovaný

*public String **decodeStringType**(ByteBuffer buffer)*

Dekóduje dáta v buffri ako reťazec v kódovaní UTF-8.

Návratová hodnota:

Reťazec v kódovaní UTF-8.

*public String **decodeOctetArrayType**(ByteBuffer buffer)*

Dáta v buffri prevedie na reťazec do kódu Base64.

Návratová hodnota:

Reťazec predstavujúci binárne dáta zakódované v Base64.

*public String **decodeDateTimeType**(String type, ByteBuffer buffer)*

Dekóduje dátové typy časových známk: dateTimeSeconds, dateTimeMilliseconds, dateTimeMicroseconds a dateTimeNanoseconds.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.

buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Referenciu reprezentujúci interpretovanú hodnotu buffra na základ predaného typu. Dátové typy `dateTimeSeconds` a `dateTimeMilliseconds` predstavujú počet sekúnd, resp. milisekúnd od Unix epochy (00:00 1.1.1970 UTC). Dátové typy `dateTimeMicroseconds` a `dateTimeNanoseconds` sú zakódované vo formáte časovej známky NTP Timestamp. Berie sa do úvahy redukované kódovanie prvých dvoch menovaných typov.

Hádže:

`DataException` - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

```
public ByteBuffer handleReducedSizeEncoding(byte[] input, int arraySize, boolean isSigned)
```

Vytvorí zo vstupného poľa bajtov buffer stanovenej dĺžky.

Parametre:

`input` - vstupné pole bajtov obsahujúce dáta skráteného informačného elementu.

`arraySize` - veľkosť informačného elementu podľa definície v informačnom modeli IPFIX.

`isSigned` - ak je dátový typ dát vo vstupnom poli bajtov znamienkové číslo, táto hodnota by mala byť `true`.

Návratová hodnota:

`ByteBuffer` obsahujúci informačný dáta informačného elementu o štandardnej veľkosti.

*public ByteBuffer **parseMacAddress**(byte[] input)*

Konvertuje bajty vo vstupnom poli bajtov na reťazec v tvare XX:XX:XX:XX:XX:XX.

Parametre:

input - vstupné pole bajtov obsahujúce dáta skráteného informačného elementu.

arraySize - veľkosť informačného elementu podľa definície v informačnom modeli IPFIX.

isSigned - ak je dátový typ dát vo vstupnom poli bajtov znamienkové číslo, táto hodnota by mala byť true.

Návratová hodnota:

ByteBuffer obsahujúci informačný dáta informačného elementu o štandardnej veľkosti.

5 Preklad programu

5.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe diplomovej práce.

Sú k dispozícii tieto zdrojové texty:

- balík sk.tuke.cnl.bm.Mediator:
 - Config.java
 - Default.java
 - DropsCounter.java
 - FlowRecordDispatcher.java
 - IPLoader.java
 - Mediator.java
 - Support.java
- balík sk.tuke.cnl.bm.Mediator.IPFIX:
 - ExporterKey.java
 - FieldSpecifier.java
 - IPFIXDataRecord.java
 - IPFIXDecoder.java
 - IPFIXElements.java
 - IPFIXEncoder.java
 - IPFIXFlowRecord.java
 - IPFIXMessage.java
 - IPFIXOptionsTemplateRecord.java
 - IPFIXSet.java
 - IPFIXTemplateRecord.java
 - IpfixUdpTemplateCache.java
 - TemplateHolder.java
- balík sk.tuke.cnl.bm.Mediator.IntermediateProcesses:
 - AIntermediateProcess.java
 - ExampleProcess.java
 - IPInputBuffer.java
- balík sk.tuke.cnl.bm.Mediator.collecting:
 - IPFIXParser.java
 - PacketCache.java
 - PacketObject.java
 - UDPProcessor.java
 - UDPServer.java
- balík sk.tuke.cnl.bm.Mediator.exporting:
 - ExportCache.java

```
MessageEncoder.java
UDPExporter.java
- balík sk.tuke.cnl.bm.exceptions:
  DataException.java
  DataFormatException.java
  EncodingException.java
  IPLoaderException.java
  MediatorException.java
  OutOfBoundsException.java
  TemplateException.java
```

5.2 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 1GB voľného miesta
- sieťová karta 100Mb/s

5.3 Požiadavky na programové prostriedky pri preklade

- operačný systém GNU/Linux s verziou jadra 2.6 a vyššou (odporúča sa kvôli podpore SCTP)
- Java Runtime Environment (JRE) verzie 1.7.0_03 a vyššej
- balík lksctp-tools
- knižnice dodávané na inštalačnom médiu

5.4 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, alebo iné implementácie.

5.5 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 obľúbený java IDE, kde stačí jednoducho nastaviť verziu JDK na 7.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam. V prostredí Netbeans IDE potom stačí kliknúť na tlačidlo *Clean and Build*.

5.6 Vytvorenie inštalačného DEB súboru

Stačí spustiť skript `buildDeb.sh`, ktorý sa nachádza v priečinku `jxcoll/deb`.

```
sh buildDeb.sh
```

Výstupom tohto skriptu je súbor s názvom `debian.deb`, ktorý môžeme následne premenovať podľa verzie JXColl (napríklad na `jxcoll_3.9_i386.deb`). Tento skript vykonáva nasledovné:

1. v prípade, ak neexistuje priečinok `debian`, extrahuje ho z archívu `debian.tar.gz`, inak tento krok preskočí
2. v prípade, ak neexistuje priečinok `debian`, extrahuje ho z archívu `debian.tar.gz`, inak tento krok preskočí

3. skopíruje binárny súbor z projektu do DEB balíčka (predpokladá sa, že bol program kompilovaný v Netbeans IDE pomocou Clean and Build tlačidla)
4. skopíruje konfiguračný súbor z projektu do DEB balíčka
5. skopíruje IPFIX definičný súbor z projektu do DEB balíčka
6. vymaže prípadné dočasné súbory z DEB balíčka
7. vygeneruje MD5 kontrolné súčty pre všetky súbory DEB balíčka
8. zabezpečí maximálnu kompresiu manuálových stránok a changelog súborov
9. skopíruje binárny súbor z projektu do DEB balíčka a nastaví mu práva na vykonávanie
10. vytvorí samotný DEB balíček
11. overí ho pomocou programu lintian - ten vypíše prípadne varovania a/alebo chyby
12. archivuje vytvorený DEB balíček do archívu debian.tar.gz

Pred spustením skriptu je nutné skompilovať JXColl pomocou Netbeans IDE tlačidlom *Clean and Build*. Prípadné zmeny control alebo changelog súboru, manuálových stránok je nutné vykonať ručne. Manuálové stránky je vhodné upraviť pomocou programu *GmanEdit* . Po spustení skriptu sa vytvorí DEB balíček s názvom `debian.deb`. Ten je vhodné premenovať podľa aktuálnej verzie. Vytvorí sa aj archív `debian.tar.gz`, ktorý obsahuje najaktuálnejšiu adresárovú štruktúru DEB balíčka pre budúce využitie (ak neexistuje priečinok `debian`, vytvorí sa extrakciou z tohto archívu). Ak je potrebné len aktualizovať kód, stačí spustiť skript a ten sa o všetko postará, pričom vytvorí aj adresár `debian`. Súbory možno v ňom upravovať až kým nie je všetko podľa predstáv. Ak je všetko hotové, v Netbeans IDE je potrebné vymazať priečinok `debian` (vykoná sa SVN DELETE, namiesto obyčajného odstránenia zo súborového systému) a projekt "commitnúť".

5.7 Opis známych chýb

V súčasnosti nie sú známe žiadne vážne chyby.

6 Zhodnotenie riešenia

Hlavným cieľom práce bolo zvýšiť interoperabilitu s inými IPFIX riešeniami pomocou zvýšenia konformity so štandardom IPFIX. V práci boli vyriešené problémy, ktoré doteraz znemožňovali dekódovanie viacerých záznamov sade, informačných elementov s variabilnou dĺžkou, informačných elementov s redukovaným kódovaním alebo niektorých predtým neimplementovaných dátových typov.

Súčasťou práce bolo rozšírenie podpory prenosu údajov o tokoch prostredníctvom transportných protokolov TCP a SCTP, čo zvyšuje možnosti nasadenia nástroja BasicMeter aj v podmienkach s vyššou náchylnosťou na preťaženie v sieti.

Možnosti budúceho vývoja zhromažďovacieho procesu nástroja BasicMeter predstavuje implementácia podpory pre dátové typy umožňujúce prenos štruktúrovaných dát a podpora pre zabezpečené pripojenia od exportérov.

Literatúra

- [1] KOBAYASHI, A. – CLAISE, B. et al.: *IP Flow Information Export (IPFIX) Mediation: Problem Statement*. RFC 5982. 2010
- [2] KOBAYASHI, A. et al.: *IP Flow Information Export (IPFIX) Mediation: Framework*. RFC 6183. 2011
- [3] SADASIVAN, G. et al.: *Architecture for IP Flow Information Export* RFC 5470. 2009
- [4] CLAISE, B.: *Cisco Systems NetFlow Services Export Version 9*. RFC 3954. 2004
- [5] VEREŠČÁK, T.: Optimalizácia zhromažďovacieho procesu nástroja BasicMeter, 2012, Diplomová práca, Príloha A, Systémová príručka JXColl v3.9, KPI FEI TU, Košice