

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	7
2	Analýza problému	7
2.1	Vybrané príklady použitia sprostredkovania správ	8
2.2	Analýza aplikačného rámca pre IPFIX Mediátor	9
3	Popis programu	11
3.1	Popis riešenia	11
4	Popis tried, členských premenných a metód	12
4.1	Balík sk.tuke.cnl.bm.Mediator	12
4.1.1	Trieda Default	12
4.1.2	Trieda DropsCounter	12
4.1.3	Trieda FlowRecordDispatcher	13
4.1.4	Trieda IPLoader	15
4.1.5	Trieda Mediator	16
4.1.6	Trieda Support	17
4.2	Balík sk.tuke.cnl.bm.Mediator.IPFIX	19
4.2.1	Trieda IPFIXEncoder	19
4.2.2	Trieda IPFIXFlowRecord	25
4.2.3	Trieda IPFIXFlowRecord	25
4.2.4	Trieda IPFIXMessage	25
4.2.5	Trieda IPFIXMessage	25
4.2.6	Trieda IPFIXTemplateRecord	25
4.3	Balik sk.tuke.cnl.bm.Mediator.collecting	26
4.3.1	Trieda UDPServer	26
4.3.2	Trieda IpfixedParser	27
5	Preklad programu	28

5.1	Zoznam zdrojových textov	28
5.2	Požiadavky na technické prostriedky pri preklade	29
5.3	Požiadavky na programové prostriedky pri preklade	29
5.4	Náväznosť na iné programové produkty	29
5.5	Vlastný preklad	30
5.6	Vytvorenie inštalačného DEB súboru	30
5.7	Opis známych chýb	31
6	Zhodnotenie riešenia	32
	Zoznam použitej literatúry	32

Zoznam obrázkov

2–1 Referenčný model sprostredkovania správ v IPFIX	9
2–2 Zjednodušený model komponentov IPFIX Mediátora	10

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*, *zokupovanie 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 *spoľahlivosť 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.

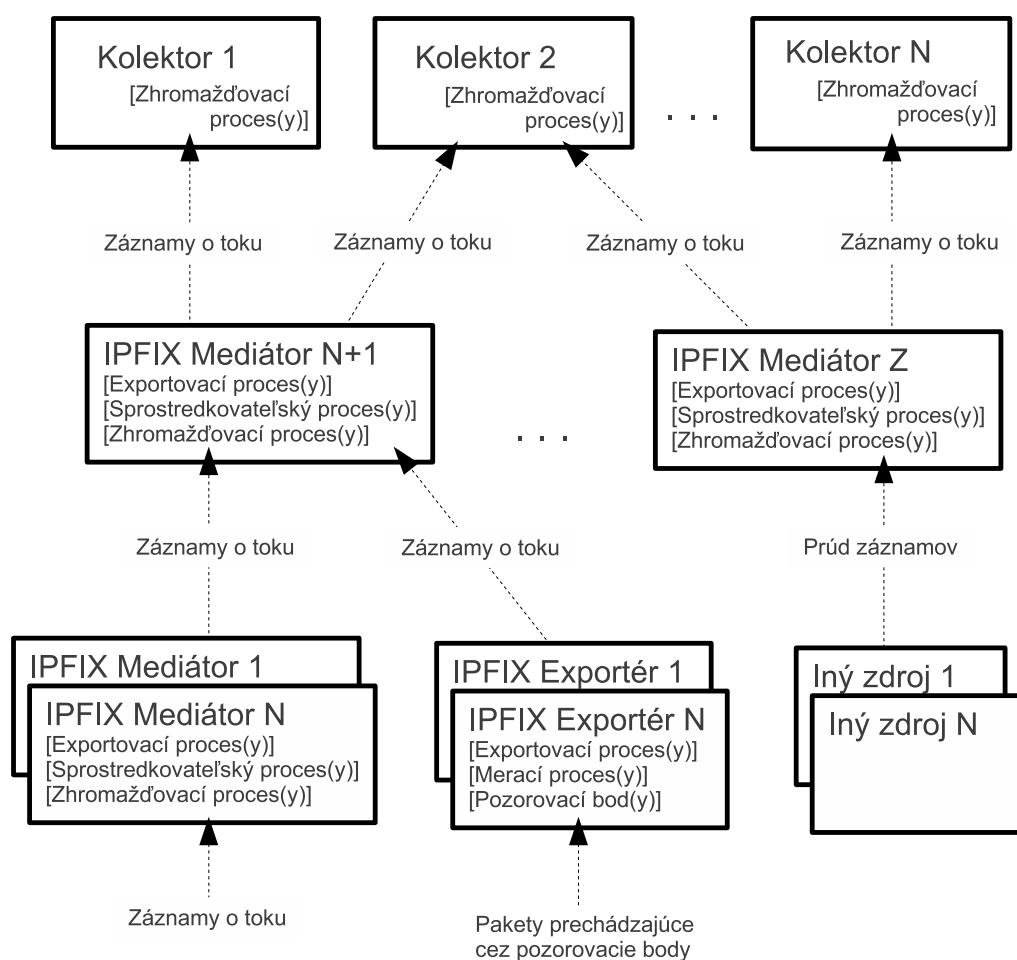
2.1 Vybrané príklady použitia sprostredkovania správ

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

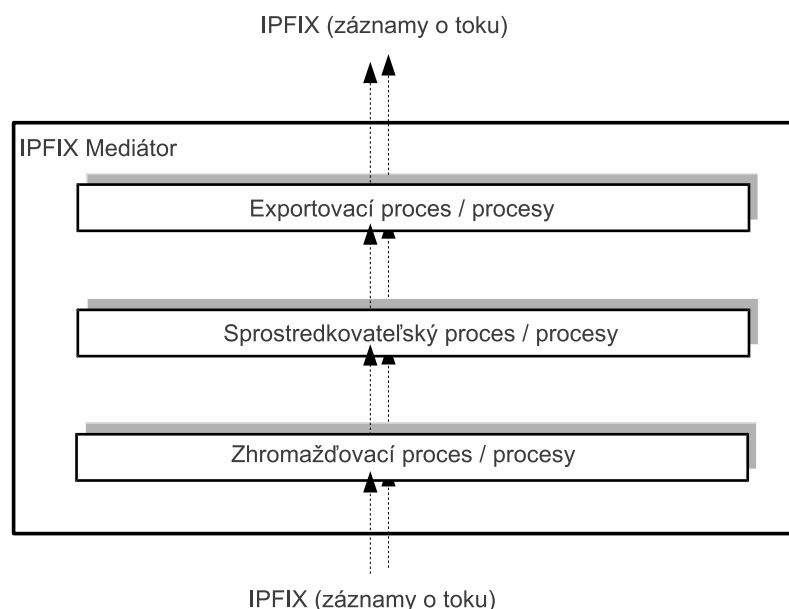
Analýze aplikačného rámca pre sprostredkovanie správ v IPFIX sa venuje RFC 6183 [2]. Na Obrázku 2–1 je zobrazený referenčný model sprostredkovania správ v IPFIX ako rozšírenie referenčného modelu IPFIX, popísaného v *Architecture for IP Flow Information Export* [3]. Táto schéma zobrazuje možné scenáre, 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.cn1.bm.Mediator.collecting` - implementacia zhromazdovacieho procesu
- `sk.tuke.cn1.bm.Mediator.exporting` - implementacia exportovacieho procesu
- `sk.tuke.cn1.bm.Mediator.IntermediateProcesses` - triedy tvoriace podporu pre sprostredkovateľské procesy. Hlavné triedy nových modulov musia byť umiestnené v tomto balíku!
- `sk.tuke.cn1.bm.Mediator.IPFIX` - triedy s manuálnou implementáciou protokolu IPFIX
- `sk.tuke.cn1.bm.Mediator.exceptions` - vlastné výnimky aplikácie
- `sk.tuke.cn1.bm.Mediator` - hlavné triedy samotného programu

3.1 Popis riešenia

4 Popis tried, členských premenných a metód

Kedže niektore triedy Mediatora su kvoli jednotnosti rieseni v ramci vyskumnej skupiny MONICA totozne s triedami nastroja JXColl, v nasledujúcich častiach budú uvedené len tie, ktoré sa tykaju vyhradne 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 vychodiskove hodnoty konfiguracneho suboru. Neobsahuje konštruktor ani žiadne metódy, iba verejne prístupne staticke konštanty.

4.1.2 Trieda `DropsCounter`

Sluzi na vypocet statistiky zahodených entit. Pod entitou sa myslia zaznamy o to-
koch, datove zaznamy, alebo IPFIX pakety. Obsahuje len staticke metódy.

Metódy

*public static void **inputBufferDropsUP()***

Zvysuje pocet strat sposobených preplnením vstupneho buffera sprostredkovateľských porcesov o jeden.

Parametre:

String processName - meno procesu

*public static void **exportCacheDropsUP()***

Zvysuje pocet strat sposobenych `ExportCache` o jeden.

```
public static void encodingDropsUp()
```

Zvysuje pocet strat sposobenych chybou pri kodovani o jeden.

```
public static void decodingDropsUp()
```

Zvysuje pocet strat sposobenych chybou pri dekodovani o jeden.

```
public static void packetDropsUp()
```

Zvysuje pocet IPFIX paketov zahodenych UDP serverom o jeden.

```
public static void printStats()
```

Vypise statistiku vsetkych zahodenych entit.

4.1.3 Trieda `FlowRecordDispatcher`

Ulohou tejto triedy je na zaklade konfiguračného suboru distribuovať prijaté zaznamy o toku sprostredkovateľským procesom (seriovo alebo paralelne) a exportovaciemu porcesu. Distribucia prebieha v súlade s IPFIX Mediator Framework (RFC 6183). Táto trieda je implementovaná podľa návrhového vzoru *Singleton*.

Metódy

```
public static FlowRecordDispatcher getInstance()
```

Implementuje vzor *Singleton*. Vytvorí a vráti jedinečnú instanciu v prípade že neexistuje, v opačnom prípade ju iba vráti.

Návratová hodnota:

Jedinecny objekt typu `FlowRecordDispatcher`.

*public synchronized void **dispatchFlowRecord**(IPFIXFlowRecord flowRecord, String inputProcess)*

Posiela prijate zaznamy o tokoch prislusnym sprostredkovatelskym procesom, alebo exportovaciemu procesu podla konfiguracie. Najprv ziska zoznam prijimatelov toku na zaklade mena povodcu. Ak je zoznam prazdny - zaznam o toku je urceny na export, preto ho zapise do `ExportCache`. Ak zoznam nie je prazdny, ziska si instance prijimatelov toku a zaznam im zapise do vstupneho buffera. Tato metoda je synchronizovana, lebo je pristupna viacerym vlaknam.

Parametre:

IPFIXFlowRecord flowRecord - zaznam o toku, ktorý sa má distribuovať ďalej

String inputProcess - meno povodcu zaznamu o toku.

*private void **fillInputBuffer**(AIntermediateProcess process, IPFIXFlowRecord flowRecord)*

Metoda, ktorá zapisuje zaznamy o tokoch do vstupneho buffra sprostredkovateľských procesov. V prípade neúspechu za zvýši počítadlo v `DropsCounter` a vypíše error.

Parametre:

AIntermediateProcess process - instance sprostredkovateľského procesu.

IPFIXFlowRecord flowRecord - zaznam o toku, ktorý sa má zapísať.

*private ArrayList<String> **getReceiversList**(String inputDevice)*

Získava zoznam príjemcov zaznamu o toku od zadaneho povodcu toku.

Parametre:

String inputDevice - povodca zaznamu o toku

Návratová hodnota:

Zoznam prijemcov toku, typ `ArrayList`.

4.1.4 Trieda `IPLoader`

Trieda je zodpovedna za dynamicke nactavanie sprostredkovatelskych procesov definovanych v konfiguracnom subore. Implementuje navrhovy vzor *Singleton*.

Metódy

public static `IPLoader` ***getInstance()***

Implementuje vzor *Singleton*. Vytvori a vrati jedinecnu instanciu v prípade že neexistuje, v opačnom prípade ju iba vrati.

public void ***loadProcesses()***

Hlavná metóda triedy, dynamicky nactáva sprostredkovateľské moduly definované v konfiguracnom subore. Najprv získa systémový *ClassLoader*. V cykle prechádza zoznam sprostredkovateľských modulov. Každý reťazec obsahujúci meno prevedie na binárne meno (meno triedy vrátane balíkov) a pomocou *ClassLoader*-a získa jeho `Class` objekt. Na základe tohto objektu získa jedinecnu instanciu modulu a keďže sa jedná o vlákno, spustí ho tak, že zavola jeho metódu `start()`.

Hádže:

`IPLoaderException` - V prípade akejkoľvek chyby, ktorá môže nastať pri vykonávaní metódy. Chyby, ktoré sú zachytávané sú typov:

- `SecurityException`
- `ClassNotFoundException`
- `IllegalAccessException`
- `NoSuchMethodException`

- `InvocationTargetException`

4.1.5 Trieda Mediator

Ulohou hlavnej triedy Mediatora je postupne spustiť všetky vlákna a procesy potrebné pre beh programu. Najprv sa precitajú a spracujú argumenty príkazového riadku. Program vie rozpoznávať dva druhy argumentov. Prvým je cesta ku konfiguračnému súboru. Ak nie je zadaná, používa sa východiskový konfiguračný súbor. Druhým argumentom môže byť zadaná možnosť `--logtofile`. Vtedy sú všetky logovacie výstupy presmerované zo štandardného výstupu do súboru.

Potom ako program načíta všetky nastavenia z konfiguračného súboru, spustí všetky svoje moduly - sprostredkovateľské procesy pomocou triedy `IPLoader`. Nasleduje spustenie vlákna, ktoré prijíma IPFIX pakety prostredníctvom protokolu UDP a vlákna, ktoré ich spracováva. Hovoríme o `UDPServer` a `UDPPProcessor`. Nakoniec je spustené exportovacie vlákno - `UDPExporter`. Kedykoľvek keď nastane chyba je Mediator korektne ukončený a to tak, že uvoľní všetku pamäť a zastaví bežiacie vlákna. Rovnako je Mediator zastavený po stlačení kombinácie kláves `Ctrl+c`.

Metódy

*public static void **main**(String[] args)*

Hlavná metóda triedy.

Parametre:

String[] args - argumenty príkazového riadku.

*public static void **stopMediator**()*

Metóda, ktorá korektne ukončuje beh programu. Zastaví všetky spustené vlákna a uvoľní všetky druhy pamäte.

*public static void **interruptThread()***

Prerusi vykonavanie vlakna.

Parametre:

Thread thread - objekt vlakna, ktore sa ma zastavit.

*private static void **loggingToFile()***

Metoda, ktora vykonava logovanie do suboru namiesto standardneho vystupu.

4.1.6 Trieda Support

Podporná trieda, ktora obsahuje pomocné metódy potrebné pri de(kodovaní) a pri validácii formátu dát. Uvádzam iba vlastné metódy.

Metódy

*public static byte[] **byteToByteArray(byte x)***

Konvertuje primitívny typ *byte* na pole bytov v usporadani bytov Big Endian.

Parametre:

byte x - hodnota, ktora sa ma zakodovat.

Návratová hodnota:

Pole bytov, typ *byte[]*.

*public static byte[] **shortToByteArray(short x)***

Konvertuje primitívny typ *short* na pole bytov v usporadani bytov Big Endian.

Parametre:

short x - hodnota, ktora sa ma zakodovat.

Návratová hodnota:

Pole bytov, typ *byte[]*.

*public static byte[] **intToByteArray**(int x)*

Konvertuje primitivny typ *int* na pole bytov v usporadani bytov Big Endian.

Parametre:

int x - hodnota, ktora sa ma zakodovat.

Návratová hodnota:

Pole bytov, typ *byte[]*.

*public static byte[] **longToByteArray**(long x)*

Konvertuje primitivny typ *long* na pole bytov v usporadani bytov Big Endian.

Parametre:

long x - hodnota, ktora sa ma zakodovat.

Návratová hodnota:

Pole bytov, typ *byte[]*.

*public static byte[] **floatToByteArray**(float x)*

Konvertuje primitivny typ *float* na pole bytov v usporadani bytov Big Endian.

Parametre:

float x - hodnota, ktora sa ma zakodovat.

Návratová hodnota:

Pole bytov, typ *byte[]*.

*public static byte[] **doubleToByteArray**(double x)*

Konvertuje primitivny typ *double* na pole bytov v usporadani bytov Big Endian.

Parametre:

double x - hodnota, ktorá sa má zakódovať.

Návratová hodnota:

Pole bytov, typ *byte[]*.

```
public static boolean validateMAC(String macAddress)
```

Validuje formát MAC adresy.

Parametre:

String macAddress - adresa, ktorej formát sa má overiť.

Návratová hodnota:

true - v prípade, že je adresa v správnom formáte.

false - opačne.

4.2 Balík *sk.tuke.cnl.bm.Mediator.IPFIX*

4.2.1 Trieda *IPFIXEncoder*

Trieda so statickými metódami slúžiacimi na zakódovanie reťazcovej reprezentácie hodnôt informacných elementov na abstraktne dátové typy podľa RFC 5101 a RFC 5102. Je presným opakom triedy *IPFIXDecoder*.

Metódy

```
public static byte[] encode(String dataType, String value)
```

Zakóduje hodnotu dátového typu do pola bytov podľa špecifikácie IPFIX. Priamo nevykonáva zakódovanie, volá konkrétne metódy podľa kategórie dátového typu.

Parametre:

String dataType - reťazec definujúci dátový typ obsiahnutý v buffri.

String value - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú retazcovu hodnotu na základ predaného typu.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`OutOfBoundsException` - Ak je hodnota mimo povoleného rozsahu

`UnknownHostException` - Ak program nevie rozpoznať host, alebo IP adresu

`DataException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

*public static byte[] **encodeUnsignedIntegralType**(String dataType, String value)*

Zakóduje celočíselné bezznamienkové dátové typy `unsigned8`, `unsigned16`, `unsigned32`, `unsigned64` a `unsigned128`.

Parametre:

String dataType - reťazec definujúci dátový typ obsiahnutý v buffri.

String value - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú retazcovu hodnotu na základ predaného typu.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`NumberFormatException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

`OutOfBoundsException` - Ak je hodnota mimo povoleného rozsahu

*public static byte[] **encodeSignedIntegralType**(String dataType, String value)*

Zakóduje celočíselné znamienkové dátové typy `signed8`, `signed16`, `signed32` a `signed64`.

Parametre:

String dataType - reťazec definujúci dátový typ obsiahnutý v buffri.

String value - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú retazcovu hodnotu na základ predaného typu.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`NumberFormatException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

*public static byte[] **encodeFloatType**(String dataType, String value)*

Zakóduje desatinné dátové typy float32 a float64.

Parametre:

String dataType - reťazec definujúci dátový typ obsiahnutý v buffri.

String value - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú retazcovu hodnotu na základ predaného typu.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`NumberFormatException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

*public static byte[] **encodeAddressType**(String dataType, String value)*

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

Parametre:

String dataType - reťazec definujúci dátový typ obsiahnutý v buffri.

String value - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú retazcovú hodnotu na základ predaného typu.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`UnknownHostException` - Ak program nevie rozpoznať host, alebo IP adresu

`DataException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

```
public static byte[] encodeBooleanType(String value)
```

Zakóduje boolean reprezentujúci pravdivostnú hodnotu.

Návratová hodnota:

Pole bytov reprezentujúci pravdivostnú hodnotu, "true" alebo "false".

Hádže:

`OutOfBoundsException` - Ak je hodnota mimo povoleného rozsahu

```
public static byte[] encodeStringType(String value)
```

Zakóduje retazec v kódovaní UTF-8 do pole bytov.

Návratová hodnota:

Pole bytov retazca v kódovaní UTF-8.

```
public static byte[] encodeOctetArrayType(String value)
```

Retazec v kódu Base64 prevedie na pole bytov.

Návratová hodnota:

Pole bytov predstavujúce binárne dáta zakódované v Base64.

```
public static byte[] encodeDateTimeType(String dataType, String value)
```

Zakóduje dátové typy časových známk: `dateTimeSeconds`, `dateTimeMilliseconds`, `dateTimeMicroseconds` a `dateTimeNanoseconds` do pola bytov.

Parametre:

String `dataType` - reťazec definujúci dátový typ obsiahnutý v buffri.

String `value` - samotné dáta, ktoré sú predmetom zakódovania

Návratová hodnota:

Pole bytov reprezentujúce interpretovanú hodnotu reťazca na základe 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.

Hádže:

`UnsupportedDataException` - Ak dátový typ nie je podporovaný

`NumberFormatException` - Ak je chyba v formate kodovanej hodnoty vzhľadom na dátový typ

`OutOfBoundsException` - Ak je hodnota mimo povoleného rozsahu

```
public static void checkStringNumbersRange(String min, String max, String value, String dataType)
```

Overi, či číselná hodnota v reťazci spadá do rozsahu daného dátovým typom.

Parametre:

String `min` - dolná hranica rozsahu

String `max` - horná hranica rozsahu

String `value` - hodnota, ktorá sa má overiť

String `dataType` - dátový typ hodnoty

Hádže:

`OutOfBoundsException` - Ak je hodnota mimo povoleného rozsahu

4.2.2 Trieda IPFIXFlowRecord

4.2.3 Trieda IPFIXFlowRecord

4.2.4 Trieda IPFIXMessage

4.2.5 Trieda IPFIXMessage

4.2.6 Trieda IPFIXTemplateRecord

————COLLECTING————

4.3 Balik sk.tuke.cnl.bm.Mediator.collecting

4.3.1 Trieda UDPServer

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

Konštruktor

```
public UDPServer()
```

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

Parametre:

int 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.

4.3.2 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. Sparované 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**.

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
 - IpfixedTemplateCache.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
- Java Runtime Environment (JRE) verzie 1.7.0_03 a vyššej
- 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 oblubený 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