Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Rozšírenie aplikačnej domény nástroja SLAmeter

Diplomová práca

Príloha E

SYSTÉMOVÁ PRÍRUČKA JXColl v4.0.1

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

Košice 2015 **Bc. Matúš Husovský**

Obsah

1 Funkcia programu **1**

2 Analýza a popis vykonaných zmien **2**

3 Opis programu **3**

3.1 Opis tried, členských premenných a metód 3

3.1.1 Trieda AccountingManager 3

3.1.2 Trieda AccountingRecord 5

3.1.3 Trieda AccountingRecordsCache 10

3.1.4 Trieda AccountingRecordsExporter 12

4 Preklad programu **13**

4.1 Zoznam zdrojových textov 13

4.2 Požiadavky na technické prostriedky pri preklade 14

4.3 Požiadavky na programové prostriedky pri preklade 15

4.4 Náväznosť na iné programové produkty 15

4.5 Vlastný preklad 15

4.6 Vytvorenie inštalačného DEB súboru 16

4.7 Opis známych chýb 17

5 Zhodnotenie **18**

6 Referencie **19**

1 Funkcia programu

Program JXColl (Java XML Collector) slúži na zachytávanie a spracovávanie informácii o tokoch v sieťach získané exportérom. Tvorí súčasť meracej architektúry BasicMeter, ktorý na základe nastavených parametrov konfiguračného súboru vie dáta získané z aktuálnej sieťovej prevádzky ukladať do databázy alebo ich sprístupniť pomocou vlastného protokolu pre priame spracovanie (protokol ACP) používateľovi. Údaje uložené v databáze sú určené 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 sieťovej hierarchie konkrétnym používateľom z hľadiska protokolov, portov, IP adries a časových charakteristík. Program bol vytvorený Ľubošom Koščom, neskôr zoptimalizovaný a doplnený novými funkciami Michalom Kaščákom, Adriánom Pekárom, Tomášom Vereščákom, Pavlom Beňkom a Marekom Marcinom.

2 Analýza a popis vykonaných zmien

Hlavným cieľom pri modifikácii programových častí tohto programu, bolo zaviesť podporu aplikačno-špecifických elementov do procesu vytvárania účtovacích záznamov. Pri vykonávaní tejto úlohy boli zistené nedostatky, ktoré sme sa snažili eliminovať.

Konkrétne boli vykonané tieto úpravy:

- Bola upravená hešovacia funkcia, ktorá v pôvodnom riešení používala len exkluzívny súčet, a umožňovala vytvárať rovnakú hodnotu hešovania pre rozličné kľúče, ktorých hodnoty boli rovnaké.
- Namiesto elementov `octetTotalCount` a `packetTotalCount`, sú v aktuálnom riešení pri vytváraní účtovacích záznamov použité elementy `octetDeltaCount` a `packetDeltaCount`.
- Do procesu tvorby účtovacích záznamov boli pridané aplikačno-špecifické elementy a tiež elementy pre zdrojovú cieľovú MAC adresu. Týmito elementami konkrétne sú:
  - applicationId
  - applicationName
  - sourceMacAddress
  - destinationMacAddress

3 Opis programu

Jednotlivé časti programu sú umiestnené v nasledujúcich balíkoch:

- sk.tuke.cnl.bm.JXColl.export - triedy určené na export údajov do databázy alebo protokolom ACP
- sk.tuke.cnl.bm.JXColl.input - triedy slúžiace na prijímanie dát z exportérov
- sk.tuke.cnl.bm.JXColl.IPFIX - triedy s manuálnou implementáciou IPFIX
- sk.tuke.cnl.bm.JXColl.accounting - triedy účtovacieho balíka
- sk.tuke.cnl.bm.JXColl - hlavné triedy samotného programu JXColl
- sk.tuke.cnl.bm.OWD - balík pre modul merania jednosmerného oneskorenia
- sk.tuke.cnl.bm - pomocné triedy a výnimky

3.1 Opis tried, členských premenných a metód

Keďže sa zmeny v tomto programe týkajú len balíka `accounting`, budú popísane len triedy tohto balíka. Ostatné triedy a ich metódy sa počas vývoja a odstraňovania chýb nezmenili a ich opis je uvedený v príručkách predošlých verzií programu.

3.1.1 Trieda AccountingManager

Na základe spracúvaných záznamov o IP tokoch, ktoré sú posielané protokolom IPFIX, generuje účtovacie záznamy a agreguje dátové toky.

Konštruktor

public ***AccountingManager****()*

Ak je v konfiguračnom súbore povolené spracovanie účtovacích záznamov, tak tento konštruktor inicializuje dočasné úložisko pre účtovacie záznamy (AccountingRe-

cordsCache) a tiež inicializuje objekt exportéra účtovacích záznamov (AccountingRecordsExporter).

Metódy

public void ***processFlow(IPFIXTemplateRecord template, IPFIXDataRecord data)****()*

Spracováva záznam o dátovom toku z IPFIX šablóny a dátového záznamu. Hodnoty z dátového záznamu sú podľa šablóny spracovávané a z nich sa vytvára alebo agreguje účtovací záznam.

- Parametre
  - `template` – šablóna
  - `data` – údaje o ip toku

public int ***generateHashKey(InetAddress srcIP, InetAddress dstIP,byte[] srcMAC,byte[] dstMAC, short protocol, int srcPort, int dstPort, short dscp, boolean multicast, int applicationId)****()*

Generuje hešovací kĺúč účtovacieho záznamu z jeho atribútov na rozuzlenie záznamov v hešovacej tabuľke. Je unikátny pre každý záznam.

- Parametre
  - `srcIP` – Atribút záznamu pre zdroj. IP adresu
  - `dstIP` – Atribút záznamu pre cieľ. IP adresu
  - `protocol` – Atribút záznamu pre transportný protokol
  - `srcPort` – Atribút záznamu pre zdroj. transportný port
  - `dstPort` – Atribút záznamu pre cieľ. transportný port
  - `dscp` – Atribút záznamu pre hodnotu DSCP
  - `multicast` – Atribút záznamu pre hodnotu multicast

- Návratová hodnota
  - `hash` – celočíselná hodnota odtlačku (int)

public static int ***byteArrayToInt(byte[] encodedValue)***
Konverzia 4bajtového poľa do celočíselnej hodnoty. Špecifická konverzia pre hodnotu identifikátora aplikácie.

- Parametre
  - `encodedValue` – ApplicationID v tvare poľa bajtov.
- Návratová hodnota
  - `value` – ApplicationID v tvare int. (int)

3.1.2 Trieda AccountingRecord

Trieda reprezentuje účtovací záznam.

Konštruktor

public ***AccountingRecord(byte[] srcIP, byte[] dstIP,byte[] srcMAC, byte[] dstMAC, short protocol, int srcPort, int dstPort, short ipdscp, long firstFlowStart,long lastFlowEnd, boolean isMulticast, long octetCount, long packetCount, int applicationId, String applicationName)***

Vytvára novú inštanciu triedy z nastavených parametrov účtovacieho záznamu.

- Parametre konštruktora
  - `srcIP` – Atribút záznamu pre zdroj. IP adresu
  - `dstIP` – Atribút záznamu pre cieľ. IP adresu
  - `srcMAC` – Atribút záznamu pre zdrojovú MAC adresu
  - `dstMAC` – Atribút záznamu pre cieľovú MAC adresu

- `protocol` – Atribút záznamu pre transportný protokol
- `srcPort` – Atribút záznamu pre zdrojový port
- `dstPort` – Atribút záznamu pre cieľový port
- `ipdscp` – Atribút záznamu pre hodnotu DSCP
- `flowTime` – Atribút záznamu pre čas exportu dátového toku
- `isMulticast` – Atribút záznamu pre príznak či sa jedná o multivastové spojenie
- `octetCount` – Atribút záznamu pre počet bajtov toku
- `packetCount` – Atribút záznamu pre počet paketov toku
- `applicationId` – Atribút záznamu pre identifikátor aplikačného protokolu pre flow
- `applicationName` – Atribút záznamu pre meno aplikačného protokolu pre flow

Metódy

public void ***addFlow(long lastFlowEnd, long octetCount, long packetCount)****()*

Pridá tok do účtovacieho záznamu. Tok musí mať rovnaké charakteristiky ako tok, z ktorého bol záznam vytvorený. K existujúcemu záznamu sa pripočíta počet bajtov a počet paketov toku.

- Parametre
  - `template` – šablóna
  - `flowTime` – čas exportu toku
  - `octetCount` – počet bajtov toku
  - `packetCount` – počet paketov toku

public boolean ***equals(Object obj)***

Porovná tento objekt s objektom v parametri. Zhodovať sa musia ip adresy, protokol, porty, dscp, multicast a MAC adresy.

- Parametre
  - `obj` – objekt na porovnanie
- Návratová hodnota
  - `value` – true, ak sú záznamy rovnaké, false inak (boolean)

public int ***hashCode()***

Vráti hash kód tohto objektu.

- Návratová hodnota
  - `hash` – hash kód

public byte[] ***getSourceIPv4Address()***

Vráti zdroj. IP adresu účt. záznamu.

- Návratová hodnota
  - `sourceIPv4Address` – zdroj. IP adresa účt. záznamu

public byte[] ***getDestinationIPv4Address()***

Vráti cieľ. IP adresu účt. záznamu.

- Návratová hodnota
  - `destinationIPv4Address` – cieľ. IP adresa účt. záznamu

public byte[] ***getSourceMACAddress()***

Vráti zdroj. MAC adresu účt. záznamu.

- Návratová hodnota
  - `sourceMAC` – zdroj. MAC adresa účt. záznamu

public byte[] ***getDestinationMACAddress()***

Vráti cieľ. MAC adresu účt. záznamu.

- Návratová hodnota
  - `destinationMAC` – cieľ. MAC adresa účt. záznamu

public short ***getProtocolIdentifier()***

Vráti trasportný protokol účtovacieho záznamu.

- Návratová hodnota
  - `protocolIdentifier` – trasportný protokol účtovacieho záznamu

public int ***getSourcePort()***

Vráti zdrojový port účtovacieho záznamu.

- Návratová hodnota
  - `sourcePort` – zdrojový port účtovacieho záznamu

public int ***getDestinationPort()***

Vráti cieľ. port účtovacieho záznamu.

- Návratová hodnota
  - `destinationPort` – cieľ. port účtovacieho záznamu

public short ***getIpDiffServCodePoint()***

Hodnota DSCP účt. záznamu.

- Návratová hodnota
  - `ipDiffServCodePoint` – hodnota DSCP účt. záznamu

public long ***getFirstFlowStart()***

Vracia čas exportu prvého toku v účtovacom zázname.

- Návratová hodnota

- `firstFlowStart` – čas exportu prvého toku v účtovacom zázname

public long ***getLastFlowEnd()***

Vracia čas kedy posledný exportovaný tok bol ukončený.

- Návratová hodnota
  - `lastFlowEnd` – čas kedy posledný exportovaný tok bol ukončený

public boolean ***isIsMulticast()***

Vracia True, ak sa jedna o multicastove spojenie, false ak nie.

- Návratová hodnota
  - `isMulticast` – True, ak sa jedna o multicastove spojenie, false ak nie

public int ***getApplicationId()***

Vracia identifikátor aplikácie účtovacieho záznamu.

- Návratová hodnota
  - `applicationId` – identifikátor aplikácie účtovacieho záznamu

public String ***getApplicationName()***

Vracia meno aplikácie účtovacieho záznamu.

- Návratová hodnota
  - `applicationName` – meno aplikácie účtovacieho záznamu

public long ***getOctetDeltaCount()***

Vracia počet bajtov účtovacieho záznamu.

- Návratová hodnota
  - `octetDeltaCount` – počet bajtov účtovacieho záznamu

public long ***getPacketDeltaCount()***

Vracia počet paketov účtovacieho záznamu.

- Návratová hodnota
  - `packetDeltaCount` – počet paketov účtovacieho záznamu

public int ***getFlowCount()***
Vracia počet dátových tokov v účtovacom zázname.

- Návratová hodnota
  - `applicationId` – počet dátových tokov v účtovacom zázname

3.1.3 Trieda AccountingRecordsCache

Trieda reprezentuje cache účtovacích záznamov.

Konštruktor

public ***AccountingRecordsCache()***

Vytvára novú inštanciu triedy a inicializuje hešovaciu tabuľku, ktorá tvorí cache účtovacích záznamov.

Metódy

public void ***addAccountingRecord(int key, AccountingRecord ar)***
Pridá účtovací záznam do cache, ak už taky neobsahuje.

- Parametre
  - `key` – kľúč účtovacieho záznamu
  - `ar` – úctovací záznam

public boolean ***containsKey(int key)***
Zistí, či sa záznam s daným kľúčom nachadza v cache

- Parametre
  - `key` – kľúč účtovacieho záznamu

- Návratová hodnota
  - `value` – true, ak sa záznam s daným kľúčom v cache nachádza, false naopak (boolean)

public AccountingRecord ***getAccountingRecord(int key)***

Vráti účtovací záznam z cache, ak sa v nej nachádza. Ináč vráti null.

- Parametre
  - `key` – kľúč účtovacieho záznamu
- Návratová hodnota
  - `value` – účtovací záznam z cache alebo null (AccountingRecord)

public void ***aggregateFlow(int key, long flowTime, long octetCount, long packetCount)***

Agreguje tok do existujúceho záznamu v cache. Tok musí mať rovnaké charakteristiky ako tok z ktorého bol účtovací záznam vytvorený.

- Parametre
  - `key` – kľúč účtovacieho záznamu, do ktorého sa tok agreguje
  - `flowTime` – čas vzniku toku
  - `octetCount` – počet bajtov v toku
  - `packetCount` – počet paketov v toku

public void ***clear()***

Vyprázdni cache.

public Hashtable<Integer, AccountingRecord> ***getArCache()***

Vráti celú hešovaciu tabuľku cache.

- Návratová hodnota

- `value` – hešovacia tabuľka cache (Hashtable<Integer, AccountingRecord>)

3.1.4 Trieda AccountingRecordsExporter

Trieda exportuje účtovacie záznamy z cache do databázy v pravidelných intervaloch.

Konštruktory

public ***AccountingRecordsExporter()***

Vytvára novú inštanciu triedy, resetuje časovač.

public ***AccountingRecordsExporter(AccountingRecordsCache cacheReference)*** Vytvára novú inštanciu triedy, resetuje časovač a predáva referenciu na cache účtovacích záznamov

- Parametre
  - `cacheReference` – referencia na cache účtovacích záznamov

Metódy

public void ***flushCacheToDB()***

Exportuje všetky účtovacie záznamy v cache do dotabázy, cache sa vyčistí.

Vnorené triedy

private ***DoExportTimerTask extends TimerTask***

Úloha pre časovač, ktorý spúšťa export do databázy.

- Metódy
  - *public void* ***run()*** – Volá metódu flushCacheToDB.

4 Preklad programu

4.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii na CD (príloha D), v umiestnení `src/slameter_collector/src`. Tiež je možné stiahnúť aktuálne zdrojové súbory zo systému GIT, príkazom:

```
git clone https://git.cnl.sk/monica/slameter_collector.git
   -c http.sslVerify=false
```

Tieto zdrojové texty sú rozdelené do nasledujúcich balíkov:

- balík sk.tuke.cnl.bm:
  - ACPIPFIXTemplate.java
  - DataException.java
  - DataFormatException.java
  - Filter.java
  - InetAddr.java
  - InvalidFilterRuleException.java
  - JXCollException.java
  - Sampling.java
  - SimpleFilter.java
  - TemplateException.java
  - Templates.java
- balík sk.tuke.cnl.bm.JXColl:
  - Config.java
  - IJXConstants.java
  - IpfixDecoder.java
  - IpfixElements.java
  - JXColl.java
  - NetConnect.java
  - PacketCache.java
  - PacketObject.java
  - RecordDispatcher.java
  - Support.java
- balík sk.tuke.cnl.bm.JXColl.export:
  - ACPServer.java
  - ACPIPFIXWorker.java
  - DBExport.java

```
   MongoClient.java
- balík sk.tuke.cnl.bm.JXColl.accounting:
   AccountingManager.java
   AccountingRecord.java
   AccountingRecordsCache.java
   AccountingRecordsExporter.java

- balík sk.tuke.cnl.bm.JXColl.accounting:
   OWDCache.java
   OWDFieldSpecifier.java
   OWDFlushCacheABThread.java
   OWDListener.java
   OWDObject.java
   OWDTemplateCache.java
   OWDTemplateRecord.java
   Synchronization.java
- balík sk.tuke.cnl.bm.JXColl.IPFIX:
   ExporterKey.java
   FieldSpecifier.java
   IPFIXDataRecord.java
   IPFIXMessage.java
   IPFIXOptionsTemplateRecord.java
   IPFIXSet.java
   IpfixSingleSessionTemplateCache.java
   IpfixUdpTemplateCache.java
   TemplateHolder.java
```

4.2 Požiadavky na technické prostriedky pri preklade

Preklad programu si vyžaduje minimálne uvedenú hardvérovú konfiguráciu:

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

4.3 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

Inštalácia databázy MongoDB, JRE a balíka lksctp-tools je uvedená v `používateľskej príručke`.

4.4 Náväznosť na iné programové produkty

Program umožňuje ukladanie prijatých dát do databázy alebo ich sprístupnenie priamym pripojením, ktoré budú následne vyhodnotené príslušnými prídavnými modulmi. Je implementáciou zhromažďovacieho procesu nástroja SLAmeter. Z toho vyplýva jeho závislosť na merací a exportovací proces - MyBeem, alebo iné implementácie.

4.5 Vlastný preklad

Preklad programu spočíva v nakopírovaní zdrojových súborov na disk a spustení kompilátora jazyka Java s potrebnými parametrami a parametrom classpath nastaveným na prídavné knižnice. Odporúča sa použiť java IDE, kde stačí jednoducho nastaviť verziu JDK na 7.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam. Vo vývojovom prostredí Netbeans IDE stačí kliknúť na tlačidlo *Clean and Build*.

4.6 Vytvorenie inštalačného DEB súboru

Vytváranie DEB balíka je možné 2 spôsobmi. Nasledujúci postup predstavuje automatizované vytvorenie. 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 môžme následne premenovať podľa verzie JXColl (napríklad na jxcoll_4.0_i386.deb). Tento skript vykonáva nasledovné operácie:

1. v prípade, ak neexistuje priečinok debian, extrahuje ho z dodávaného archívu debian.tar.gz, inak tento krok preskočí
2. skopíruje binárny súbor aplikácie z projektu do DEB balíčka (predpokladá sa, že bol program kompilovaný v Netbeans IDE pomocou Clean and Build tlačidla)
3. skopíruje konfiguračný súbor aplikácie z projektu do DEB balíčka
4. skopíruje IPFIX definičný súbor aplikácie z projektu do DEB balíčka
5. vymaže prípadné dočasné súbory nachádzajúce sa v DEB balíčku
6. vygeneruje MD5 kontrolné súčty pre všetky súbory DEB balíčka
7. zabezpečí maximálnu kompresiu manuálových stránok a changelog súborov
8. skopíruje binárny súbor z projektu aplikácie do DEB balíčka a nastaví mu práva na vykonávanie
9. vytvorí samotný DEB balíček
10. overí ho pomocou programu lintian - ten vypíše prípadne varovania a/alebo chyby ktoré je následne potrebné manuálne odstrániť
11. 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 je automaticky vytvorený DEB balíček s názvom `debian.deb`. Ten je vhodné premenovať podľa aktuálnej verzie pre zachovanie prehľadnosti. 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 všetky potrebné náležitosti postará, pričom vytvorí aj adresár debian. Súbory je v ňom možno 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".

4.7 Opis známych chýb

V súčasnosti nie sú známe žiadne vážne chyby.

5 Zhodnotenie

Program JXCOll je pripravený na použitie. Ponúka možnosť zachytávania a spracovávania informácií o tokoch v sieťach. Informácie získané od Exportérov je schopný exportovať do databázy na základe údajov z konfiguračného súbou. Taktiež je schopný údaje exportovať priamo používateľovi využijúc vlastný protokol pre priame spracovanie (protokol ACP). V tejto verzii podporuje príjem aplikačno-špecifických elementov z exportéra a ich ukladanie do databázy.

6 Referencie

[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

[3] Pekár, A.: Meranie prevádzkových parametrov siete v reálnom čase, 2009, Bakalár-ska práca, KPI FEI TU, Košice

[4] Vereščák, T.: Zhromažďovací proces nástroja BasicMeter, 2010, Bakalárska práca, KPI FEI TU, Košice

[5] Pekár, A.: Optimalizácia zhromažďovacieho procesu nástroja BasicMeter, 2011, Diplomová práca, KPI FEI TU, Košice

[6] Vereščák, T.: Optimalizácia zhromažďovacieho procesu nástroja BasicMeter, 2012, Diplomová práca, KPI FEI TU, Košice

[7] Benko, P.: Aplikačné rozhranie pre vyhodnotenie sieťovej prevádzky v reálnom čase, 2013, Bakalárska práca, KPI FEI TU, Košice

[8] Marcin, M.: Vyhodnocovanie prevádzkových parametrov počítačových sietí, 2015, Diplomová práca, KPI FEI TU, Košice