Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Vyhodnocovanie prevádzkových parametrov počítačových sietí

Diplomová práca

Príloha C

SYSTÉMOVÁ PRÍRUČKA JXColl v4.0

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

Košice 2014 **Bc. Marek Marcin**

Obsah

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 a Pavlom Beňkom.

2 Analýza riešenia

Jedným z prostriedkov optimalizácie nástroja SLAmeter bola zmena texhnológie databázy. Dôvodom bolo zlepšenie komunikácie s Vyhodnocovačom a podpora rôznych agregačných funkcií, ktoré technológia MongoDB poskytuje. Sú to napríklad agregačné rúry, či funkcia MapReduce.

Zmenu texhnológie databázy je samozrejme potrebné aplikovať aj na nástroj JXColl. Aby sa IPFIX údaje z Exportéra ďalej ukladali do novej databázy. V prvom rade je potrebné zabezpečiť spojenie s databázou MongoDB vytvorením triedy na to určenej a taktiež zmenou existujúcich tried tak, aby bola zabezpečená správna funkčnosť nástroja. Úprava sa premiente aj v konfiguračnom súbore *jxcoll_config.xml*.

Najdôležitejšou časťou bude upraviť všetky triedy zabezpečujúce export údajov z Kolektora do databázového Úložiska. To predstavuje zmenu základnej exportovacej triedy *RedordDispatcher.java* a tried na ňu naväzujúcich, ako aj tried starajúcich sa o export údajov v balíku *OWD*, pre meranie jednosmerného oneskorenia.

Je potrebné si uvedomiť, že MongoDB databáza neprijíma SQL príkazy ako to bolo u PostgreSQL, ale BSON správy. V programovacom jazyku Java sa pre komunikáciu využívajú knižnice na to určené. Hodnoty sa do databázy neexportujú v textovom tvare, ale ako hodnoty adekvátnych typov. Z toho dôvodu je potrebná zmena exportovaných dátových typov. Tá sa zabezpečí úpravou triedy *IpfixDecoder.java* tak, aby exportované dátové typy boli podporované MongDB databázou a zároveň, aby sa zachovala konformita s IPFIX štandardom.

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 modulu
- 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

Kedže sa počas vývoja a odstraňovania chýb pôvodné triedy a ich metódy nezmenili, v nasledujúcich častiach budú uvedené len tie, ktoré boli pridané a modifikované počas riešenia tejto diplomovej práce. Opis ostatných tried a metód je uvedený v príručkách predošlých verzií programu.

3.1.1 Trieda RecordDispatcher

Trieda predstavuje systém parsovania hodnôt pre databázu, ACP a pre účtovanie.

Konštruktor

public ***RecordDispatcher****()*

Konštruktor slúži na inicializovanie objektov databázy, ACP a účtovania. Konštruktor je singleton, čo značí že inicializácia objektu prebehne jeden krát.

Metódy

public static ***RecordDispatcher****()*

Metóda slúži na prístup k singleton objektu danej triedy.

Návratová hodnota:

Singleton objekt triedy RecordDispatcher.

public synchronized void ***dispatchIPFIXRecord****(IPFIXTemplateRecord template,*
IPFIXDataRecord data,
InetSocketAddress ipmb)

Odovzdá prijaté dáta a im zodpovedajúcu šablónu všetkým aktívnym modulom kolektora, ktoré pracujú s IPFIX správami.

Parametre:

template – šablona obsahujúca dáta.

data - dátový záznam.

ipmb - adresa.

public void ***dbExport****(IPFIXTemplateRecord template,*
IPFIXDataRecord dataRecord)

Metóda slúži na export všetkých nameraných dát poslaných protokolom IPFIX do databazy a uloženie do hashtable pre ACP. Pri prvom prechode funkciou sa generuje pamäťový záznam o informačných elementoch (IE) z XML súboru. Získajú sa informácie o IE, ktoré sa nachádzajú v šablóne, dekódujú sa ich dátové typy a príslušnosť k skupine pre uloženie hodnôt do databázy.

Parametre:

template – šablóna obsahujúca dáta.

dataRecord – dátový záznam.

public void ***ParseForACP****(IPFIXTemplateRecord template,*
IPFIXDataRecord dataRecord)

Metóda slúži na uloženie nameraných hodnôt do hašovacej tabuľky v prípade, že je vypnutý export pre databázu.

Parametre:

template – šablóna obsahujúca dáta.

dataRecord – dátový záznam.

public void ***closeDBConnection****()*

Metóda slúži na korektné uzatvorenie spojenia s databázou.

public Hashtable ***getData****()*

Metóda slúži na prístup k hašovacej tabuľke, ktorá obsahuje názov IE ako kľúč a hodnotu predstavujúcu konkrétnu nameranú hodnotu.

Návratová hodnota:

Vráti hašovaciu tabuľku obsahujúcu dáta.

3.1.2 Trieda IpfixDecoder

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

Metódy

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

Dekóduje dátový typ obsiahnutý v buffri do podoby reťazca podľa špecifikácie IPFIX. Priamo nevykonáva dekódovanie, volá konkrétne metódy podľa kategórie

dátového typu.

Parametre:

type – reťazec definujúci dátový typ obsiahnutý v buffri,

buffer – samotné dáta, ktoré sú predmetom dekódovania.

Návratová hodnota:

Objekt reprezentujúci interpretovanú hodnotu buffra na základe predaného typu.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

public Object ***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 dekódovania.

Návratová hodnota:

Objekt reprezentujúci interpretovanú hodnotu buffra na základe predaného typu.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

public Object ***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:

Objekt reprezentujúci interpretovanú hodnotu buffra na základe predaného typu.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

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

Dekóduje 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:

Objekt reprezentujúci interpretovanú hodnotu buffra na základe predaného typu.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

public Object ***decodeAdressType****(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 dekódovania.

Návratová hodnota:

Objekt reprezentujúci interpretovanú hodnotu buffra na základe predaného typu.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

public Object ***decodeBooleanType****(ByteBuffer buffer)*

Dekóduje boolean reprezentujúci pravdivostnú hodnotu.

Parametre:

buffer – samotné dáta, ktoré sú predmetom dekódovania.

Návratová hodnota:

Jeden z objektov *BOOLEAN.True* alebo *BOOLEAN.False* reprezentujúce pravdivostnú hodnotu, ”true” alebo ”false”.

Výnimky:

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

public Object ***decodeStringType****(ByteBuffer buffer)*

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

Parametre:

buffer – samotné dáta, ktoré sú predmetom dekódovania.

Návratová hodnota:

Reťazec v kódovaní UTF-8.

public Object ***decodeOctetArrayType****(ByteBuffer buffer)*

Dáta v buffri prevedie na bajtové pole.

Parametre:

buffer – samotné dáta, ktoré sú predmetom dekódovania.

Návratová hodnota:

Bajtové pole.

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

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

Parametre:

type – reťazec definujúci dátový typ obsiahnutý v buffri,

buffer – samotné dáta, ktoré sú predmetom dekódovania.

Návratová hodnota:

Objekt 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.

Výnimky:

UnsupportedDataTypeException – Ak dátový typ nie je podporovaný.

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

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.

Návratová hodnota:

Reťazec v tvare XX:XX:XX:XX:XX:XX, reprezentujúci MAC adresu.

Výnimky:

DataException – Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ.

3.1.3 Trieda OWDCache

Trieda funguje ako medzi-zásobník v ktorom sa údaje držia po dobu nastavenú v konfiguračnom súbore JXColl. Obsahuje dva zásobníky, každý pre daný merací bod medzi ktorými sa hľadá jednosmerné oneskorenie. Trieda tiež obsahuje metódy na identifikáciu zodpovedajúcich časových známok, na základe ktorých sa určuje hodnota owd. V triede boli vytvorené pomocné metódy pre prácu so zásobníkmi.

Konštruktor

public ***OWDCache****()*

Vytvára novú inštanciu triedy. Vytvára spojenie s databázou na export hodnôt owd.

Metódy

public void ***pushA****(long timeStamp, short protocolIdentifier,*

int sourceTransportPort,
Inet4Address sourceIPv4Address,
int destinationTransportPort,
Inet4Address destinationIPv4Address,
short ipVersion, long observationPointID,
long flowID, BigInteger flowStart,
BigInteger flowEnd, byte[] firstPacketID,
byte[] lastPacketID, byte[] packet,
InetAddress addr)

Metóda určuje hodnoty owd. Keď zásobník A je prázdny, objekt sa vloží do zásobníka B. Keď zásobník A nie je prázdny, objekt sa podľa filtračných kritérií porovná so všetkými objektmi v zásobníku A. Keď kritéria sú splnené, vypočíta sa hodnota owd a zapíše sa do databázy.

Parametre:

timeStamp – long čas zapísania objektu do zásobníka,
protocolIdentifier – short protocolIdentifier,
sourceTransportPort – int sourceTransportPort,
sourceIPv4Address – Inet4Address sourceIPv4Address,
destinationTransportPort – int destinationTransportPort,
destinationIPv4Address – Inet4Address destinationIPv4Address,
ipVersion – short ipVersion,
observationPointID – long observationPointID,
flowID – long flowID,
flowStart – BigInteger flowStartNanoseconds,
flowEnd – BigInteger flowEndNanoseconds,
firstpacketID – byte[] firstPacketID,
lastPacketID – byte[] lastPacketID,
packet – byte[] údaje z paketu,

addr – InetAddress adresa od ktorej paket prišiel.

Výnimky:

InterruptedException – pri prerušení počas čakania na zápis do zásobníka,

NullPointerException – pri *null* hodnote na vstupe.

public void ***pushB****(long timeStamp, short protocolIdentifier,*
int sourceTransportPort,
Inet4Address sourceIPv4Address,
int destinationTransportPort,
Inet4Address destinationIPv4Address,
short ipVersion, long observationPointID,
long flowID, BigInteger flowStart,
BigInteger flowEnd, byte[] firstPacketID,
byte[] lastPacketID, byte[] packet,
InetAddress addr)

Metóda určuje hodnoty owd. Keď zásobník B je prázdny, objekt sa vloží do zásobníka A. Keď zásobník B nie je prázdny, objekt sa podľa filtračných kritérií porovná so všetkými objektmi v zásobníku A. Keď kritéria sú splnené, vypočíta sa hodnota owd a zapíše sa do databázy.

Parametre:

timeStamp – long čas zapísania objektu do zásobníka,

protocolIdentifier – short protocolIdentifier,

sourceTransportPort – int sourceTransportPort,

sourceIPv4Address – Inet4Address sourceIPv4Address,

destinationTransportPort – int destinationTransportPort,

destinationIPv4Address – Inet4Address destinationIPv4Address,

ipVersion – short ipVersion,

observationPointID – long observationPointID,

flowID – long flowID,

flowStart – BigInteger flowStartNanoseconds,

flowEnd – BigInteger flowEndNanoseconds,

firstpacketID – byte[] firstPacketID,

lastPacketID – byte[] lastPacketID,

packet – byte[] údaje z paketu,

addr – InetAddress adresa od ktorej paket prišiel.

Výnimky:

InterruptedException – pri prerušení počas čakania na zápis do zásobníka,

NullPointerException – pri *null* hodnote na vstupe.

public static boolean ***compareFirstPacketID****(OWDObject a, OWDObject b)*

Metóda ktorá porovná dva objekty na základe ich firstPacketID.

Parametre:

a – OWDObject objekt A,

b – OWDObject objekt B.

Návratová hodnota:

true – keď objekty sú identické,

false – v opačnom prípade.

public static boolean ***compareLastPacketID****(OWDObject a, OWDObject b)*

Metóda ktorá porovná dva objekty na základe ich lastPacketID.

Parametre:

a – OWDObject objekt A,

b – OWDObject objekt B.

Návratová hodnota:

true – keď objekty sú identické,

false – v opačnom prípade.

public static boolean ***compareKeys****(OWDObject a, OWDObject b)*

Metóda ktorá porovná niektoré informačné elementy dvoch objektov.

Parametre:

a – OWDObject objekt A,

b – OWDObject objekt B.

Návratová hodnota:

true – keď dané informačné elementy objektov sa zhodujú,

false – v opačnom prípade.

public static String ***getHex****(byte[] raw)*

Metóda ktorá vráti hexadecimálnu reprezentáciu objektu v podobe pole bajtov.

Parametre:

raw – byte[] hodnota v podobe pole bajtov.

Návratová hodnota:

textová reprezentácia hexadecimálnej hodnoty.

public static void ***closeDBConnection****()*

Metóda ukončujúca spojenie s databázou.

public static int ***getNumberOfElementsA****()*

Vráti počet objektov v zásobníku A.

Návratová hodnota:

počet objektov v zásobníku A typu *int*.

public static int ***getNumberOfElementsB****()*

Vráti počet objektov v zásobníku B.

Návratová hodnota:

počet objektov v zásobníku B typu *int*.

public statit OWDObject ***pullA****(int i)*

Odstráni objekt zo zásobníka A na požadovanej pozícii.

Parametre:

i – int pozícia objektu, ktorý sa má odstrániť.

Návratová hodnota:

OWDObject objekt, ktorý sa odstránil.

Výnimky:

InterruptedException – keď nastalo prerušenie počas čakania na zápis.

public statit OWDObject ***pullB****(int i)*

Odstráni objekt zo zásobníka B na požadovanej pozícii.

Parametre:

i – int pozícia objektu, ktorý sa má odstrániť.

Návratová hodnota:

OWDObject objekt, ktorý sa odstránil.

Výnimky:

InterruptedException – keď nastalo prerušenie počas čakania na zápis.

public statit OWDObject ***readA****(int i)*

Vráti objekt zo zásobníka A na požadovanej pozícii.

Parametre:

i – pozícia objektu typu *int*.

Návratová hodnota:

OWDObject objekt na danej pozícii.

Výnimky:

InterruptedException – keď nastalo prerušenie počas čakania na zápis.

public statit OWDObject ***readB****(int i)*

Vráti objekt zo zásobníka B na požadovanej pozícii.

Parametre:

i – pozícia objektu typu *int*.

Návratová hodnota:

OWDObject objekt na danej pozícii.

Výnimky:

InterruptedException – keď nastalo prerušenie počas čakania na zápis.

public void ***setlogl****(String level)*

Nastavuje úroveň logovania.

3.1.4 Trieda AccountingRecordsExporter

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

Konštruktory

public ***AccountingRecordsExporter****()*

Vytvára novú inštanciu triedy a zároveň resetuje časovač.

public ***AccountingRecordsExporter****(AccountingRecordsCache cacheReference)*

Vytvára novú inštanciu triedy, resetuje časovač a predáva referenciu na cache učtovacích záznamov.

Metódy

public void ***flushCacheToDB****()*

Exportuje všetky účtovacie záznamy z cache do databázy. Cache sa vyčistí.

3.1.5 Trieda DBExport

Abstraktná trieda pre prístup k databáze MongoDB.

Konštruktor

public ***DBExport****()*

Vytvára novú inštanciu triedy.

Metódy

public boolean ***isConnected****()*

Kontrola spojenia s databázou.

Návratová hodnota:

true – ak spojenie je nadviazané,

false – v opačnom prípade.

public void ***connect****(String host, String port,*
String name,
String username, String password)

Metóda vytvára spojenie s databázou MongoDB. Ak sa nepodarí nadviazať spojenie, ukončí sa exportovací proces a tým aj ďalší beh aplíkácie JXColl. **Parametre:**
host – IP adresa, alebo názov databázového servra,
port – port databázového servra,
name – meno databázy,
username – prihlasovacie meno,
password – prihlasovacie heslo.

public void ***disconnect****()*

Metóda pre ukončenie spojenia s databázou.

public DB ***getDb****()*

Datbázový getter.
Návratová hodnota:
Objekt predstavujúci databázové spojenie.

3.1.6 Trieda MongoClient

Implementácia prístupu k MongoDB databáze. Trieda dedí od abstraktnej triedy *DBExport*.

Konštruktor

public ***MongoClient****()*

Vytvára novú inštanciu triedy.

Metódy

public void ***dbConnect****()*

Metóda pre vytvorenie spojenia s databázou.

public void ***insertdata****(String tabname, String json)*

Metóda pre vloženie dát do databázy.

Parametre:

tabname – meno databázovej kolekcie,

json – dátam ktoré sa vložia do databázy.

public void ***insertdata****(String tabname, String[] columns, Object[] values)*

Metóda pre vloženie dát do databázy. Metóda z názvu stĺpcov (v MongoDB databáze sú to polia, angl. fields) a hodnôt vytvorí databázový objekt, ktorý vloží do tatabázy.

Parametre:

tabname – meno databázovej kolekcie,

columns – mená polí v kolekcii,

values – hodnoty.

public double ***getNextSequenceNumber****(String seq_name)*

Metóda vracia nasledujúce sekvenčné číslo požadovanej sekvencie.

Parametre:

seq_name – meno sekvencie.

Návratová hodnota:

sekvenčné číslo.

public double ***getCurrentSequenceNumber****(String seq_name)*

Metóda vracia aktuálne sekvenčné číslo požadovanej sekvencie.

Parametre:

seq_name – meno sekvencie.

Návratová hodnota:

sekvenčné číslo.

4 Preklad programu

4.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe A CD, časť JXColl bakalárskej práce.

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

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 najaktuálnej-

šiu 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).

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

[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