Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra počítačov a informatiky

Monitorovanie prevádzkových parametrov siete v reálnom čase

Bakalárska práca

Príloha D

SYSTÉMOVÁ PRÍRUČKA (JXColl v.3)

Adrián Pekár

Vedúci práce: Ing. Juraj Giertl, PhD.

Konzultant práce: Ing. Martin Révés

Košice 2009

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Funkcia programu

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

2 Analýza riešenia

Keďže sa jedná o rozšírenie existujúceho riešenia, prostriedky vývoja sa nemenili. Koncepcia a štruktúra Kolektora sa zachovala, pričom sa najväčšej zmene podrobilo priame pripojenie, ktoré bolo nahradené podporou protokolu ACP.

Analyzer Collector Protocol (ACP) je binárny protokol aplikačnej vrstvy, ktorý slúži na sprístupnenie dát používateľovi pre priame spracovanie a pomocou ktorého sa monitorovanie prevádzkových parametrov sieťovej prevádzky môže uskutočniť v takmer reálnom čase. Pri používaní protokolu ACP treba rátať s istým oneskorením, ktorá vyplýva zo spracovania údajov jednotlivými vrstvami architektúry BasicMeter.

Komunikácia je založená na posielaní žiadostí Analyzérom a posielaní odpovedí a dát Kolektorom, pričom Analyzér predstavuje aplikácia, ktorá má v sebe implementované aplikačné rozhranie pre obsluhu protokolu ACP (ACPapi), a Kolektor samotný program JXColl s podporou protokolu ACP. Každá správa počas komunikácie má svoj jedinečný identifikátor (id), pričom celým protokolom sa posielajú:

- (0) - dáta
- (1) - riadiace správy

Riadiace správy prijaté od Analyzéra predstavujú žiadosti na Kolektor a sú nasledovné:

- nesprávny (správny) autentifikačný údaj (identifikátor správy A)
- nesprávna (správna) šablóna (0)
- nesprávny (správny) filter (1)
- nesprávne (správne) prerušenie posielania dát (2)
- nesprávna (správna) obnova posielania dát (3)
- nepodporovaný (podporovaný) typ prenosu (4)
- potvrdenie prijatia dát (5)

Autentifikačné údaje slúžia na nadviazanie spojenia s Kolektorom. Šablóna a filter slúžia na nastavenie formy a rozsah dát, v ktorom ich Analyzér požaduje, pričom samotný prenos týchto dát je možne ľubovolne pozastaviť alebo obnoviť pomocou ďalších dvoch riadiacich správ. Typ prenosu je možné dvomi spôsobmi, a to buď po jednom alebo po n-ticiach, kde n predstavuje počet identifikátorov informačných elementov v šablóne. Posledná riadiaca správa slúži na potvrdenie prijatých dát Analyzérom, a teda reakcia na túto správu spočíva v následnom poslaní ďalších dát.
Filtračné kritéria protokolu ACP sú:

- merací bod
- zdrojová a cieľová IP adresa
- zdrojový a cieľový port
- protokol

Z pohľadu systémovej príručky, ďalšie podrobnosti o protokole ACP nie sú potrebné. Jeho podrobná dokumentácia pre prípad potreby je však zahrnutá medzi príloh bakalárskej práce (viď. Príloha E, Špecifikácia protokolu ACP).

3 Zmeny v Kolektore oproti starej verzii (JXColl2)

Počas vývoja podpory pre priame pripojenie protokolom ACP sa zistilo viacero nedostatkov funkčnosti Kolektora. Po zhodnotení týchto nedostatkov sa JXColl podrobil niekoľkým zmenám. Prvá a najpodstatnejšia zmena bola už vyššie spomenuté prerobenie podpory pre priame pripojenie. Síce predošlá verzia takúto podporu už poskytovala, ale iba pre protokoly NetFlow verzie 5 a 9. Naviac, stará verzia nepodporovala každú riadiacu správu prijatú cez priame pripojenie (napr. pozastavenie alebo obnova prenosu dát). Nová verzia Kolektora (verzia 3.1) už plne vyhovuje požiadavkám protokolu ACP.

Kvôli chýbajúcemu NetFlow Exportéru, export NetFlow údajov do databázy a starým protokolom DC neboli overené. Z toho dôvodu zdrojové kódy pre spracovanie týchto údajov boli vybraté z programu, ktoré však v prípade potreby môžu byť kedykoľvek obnovené.

Ďalším zmenám sa podrobilo sparsovanie IPFIX paketov prijatých od Exportéra, v ktorom kvôli chybnému návrhu vytvárania IPFIX údajov došlo k preplneniu pamäte. Táto závažná chyba spôsobila po istej dobe zamrznutie Kolektora a následne pád celého BasicMetra. Po upravení tejto chyby, zamrznutie ale naďalej pretrvávala. Po dôkladnej analýze využitia pamäte sa preukázalo, že extrémne zahltenie pamäte JVM sa vyskytuje pri väčšej sieťovej prevádzke. Celá štruktúra exportovania IPFIX údajov (buď do databázy alebo na priame pripojenie) je založená na XML súbore, ktorá sa po prijatí nového paketu prehľadáva znova a znova. Takýmto spôsobom sa priraďujú údaje k príslušným informačným elementom pre ďalšie spracovanie. Po prijatí väčšieho množstva údajov je výkon počítača (procesor a voľná pamäť) už nepostačujúci, a tým dochádza k nakopeniu objektov, ktoré čakajú na spracovanie v java kope (java heap). Toto nakopenie po istej dobe tiež spôsobuje zamrznutie Kolektora, ale životnosť JXColl je už oveľa dlhšia ako v

predošlej verzii, navyše pri slabej alebo vyváženej prevádzke by k pádu vôbec nemalo dojsť.

Táto chyba sa prepísaním niektorých časti programu nedá odstrániť, a preto sa v blízkej budúcnosti predpokladá väčšia zmena architektúry spracovania IPFIX paketov.

Šablóny prijaté Exportérom, obsahujúce informačné elementy vytvorené v rámci podprojektov laboratória CNL spôsobovali nesprávne pracovanie celého programu. Chybu spôsoboval preklep v zdrojovom kóde, ktorá viedla k nesprávnemu rozpoznaniu hodnôt s pravdivým enterprise bitom. Chyba bola odstránená a následne boli pridané ďalšie informačné elementy do vyššie spomínaného XML súboru, a zároveň bola pre nich vytvorená tabuľka v databázovom skripte (skript sa nachádza na inštalačnom médiu bakalárskej práce, príloha F). Zoznam týchto informačných elementov sú zahrnuté v tabuľke 3 – 1. Ich podrobné popisy sú uvedené v prácach Mariána Smoleja a Slavomíra Strhárskeho.

Kvôli vyššie uvedeným problémom s pamäťou, XML súbor (ipfixFileds.xml), obsahujúci informácie potrebné na spracovanie IPFIX údajov, sa úplne vyčistil. V súčasnom stave *ipfixFields.xml* súbor obsahuje informácie iba o tých elementoch, ktoré sú podporované aj samotným exportovacím procesom, teda Exportérom (BEEM). Zoznam týchto elementov je uvedený v tabuľke 4 – 1. Samozrejme informačné elementy CNL sú naďalej podporované a pre prípad potreby sa na inštalačnom médiu (príloha F) nachádza aj pôvodný súbor pod názvom *ipfixFieldsOLD.xml*.

Posledné zmeny predstavujú pridanie podpory vypínania a zapínania exportu účtovacích záznamov do databázy a nastavenie času, po ktorom sa šablóna pre IPFIX paket považuje za neplatnú. Tieto voľby boli doteraz pevne zadefinované a ich zmena bola možná iba manuálnym prepísaním istých časti zdrojových kódov. V novej verzii JXColl je nastavenie týchto hodnôt už pridaná do *JXColl.conf* súboru, medzi ostatné nastavenia funkčnosti Kolektora.

Tabuľka 3 – 1 Informačné elementy podporované Kolektorom

Názov	Identifikátor	Enterprise id	Popis
roundTripTimeNanoseconds	240	26235	reprezentuje čas medzi zachytením dvoch paketov v pozorovanom bode pre príslušný tok, ktoré tvoria takzvaný paketový pár
packetPairsTotalCount	241	26235	reprezentuje celkový počet paketových párov identifikovaných pre príslušný tok
hostnameOrIP	250	26235	IP adresa alebo názov hostu používateľa
logCount	251	26235	počet záznamov - logov v exportovanom toku
objectsSize	252	26235	koľko bytov stiahol daný tok (používateľ) zo servera
userBrowser	253	26235	používateľov prehliadač
operationSystem	254	26235	používateľov operačný systém

4 Popis programu

Jednotlivé časti programu sú umiestnené v nasledujúcich balíkoch:

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

4.1 Popis riešenia

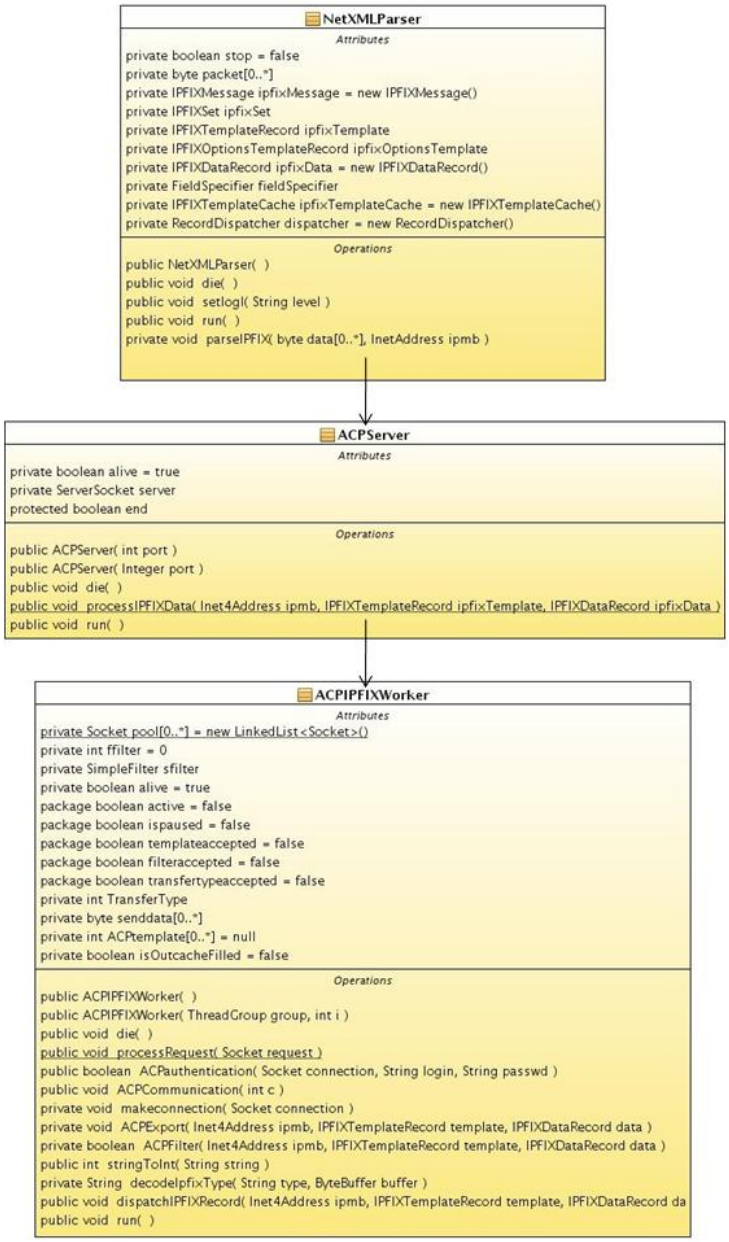
Spracovanie IPFIX dátových záznamov sa deje podľa presne špecifikovanej šablóny. Táto šablóna nariaďuje odchytávať a exportovať tie charakteristiky sieťovej prevádzky, ktoré vyhovujú filtračným kritériám a nastavenej šablóne protokolu ACP. Celý program je zložený zo samostatne fungujúcich a kooperujúcich vlákien, pričom pre vlákna obsluhujúce spojenia pre priame pripojenie na Kolektor je vytvorená skupina piatich vlákien.

Triedy, cez ktoré prebieha celý proces od prijatia údajov od Exportéra až do ich exportovania cez priame pripojenie, sú uvedené na obrázku 4 – 1.

NetXMLParser je hlavnou triedou spracovania dát. Konzumuje dáta uložené vo vyrovnávacej pamäti, ktoré sú neskôr prostredníctvom metódy parseIPFIX() posunuté buď pre priame pripojenie (ACP) alebo pre ukladanie do databázy. Trieda ACPServer po obdržaní sparsovanej šablóny s príslušnými dátami skontroluje stavy pripojenia cez protokol ACP. Ak ACPServer nezistí žiadne žive spojenie objekty zahodí, ale v opačnom prípade ich posuva na ďalšie spracovanie triedou ACPIPFIXWorker.

V triede ACPIPFIXWorker po obdržaní každého objektu sa najprv skontroluje stav zásobníka, z ktorého sa posielajú hodnoty. Kým sa zásobník nevyprázdni spracovanie nových údajov sa neuskutoční. Keď zásobník je prázdny a prenos nie je pozastavený Analyzujúcou aplikáciou, objekty sa posúvajú metóde, v ktorej prebieha prirovnanie

paketu filtračným kritériám, a v prípade zhody sa hodnoty podľa prijatej šablóny pripravia na prenos. Pritom sa neustále monitoruje príchod riadiacich správ ktorým podľa možností JXColl buď vyhovie, alebo ich odmietne. Kontrola údajov podľa nastavených filtračných kritérií a šablón prebieha pomocou tried spoločného balíku sk.tuke.cnl.bm.



Obrázok 4 – 1 Triedy spracúvajúce údaje pre protokol ACP

4.2 Popis tried, členských premenných a metód

Kedže sa počas vývoja a odstraňovania chýb pôvodné triedy a ich metódy nezmenili, v nasledujúcich častiach budú uvedené len tie, ktoré sú z implementačného pohľadu podpory protokolu ACP v programe JXColl potrebné. Popis ostatných tried a metód je uvedený v príručkach predošlých verzií programu.

4.2.1 Trieda ACPServer

Trieda slúži na vytváranie a správu vlákien pre priame pripojenie. Predáva IP adresu meracieho bodu, IPFIX šablónu a údaj pre spracovanie protokolom ACP.

Koštruktor

public **ACPServer**(int port)
throws java.io.IOException
Vytvorí ACP vlákno, ktoré bude čakať na TCP pripojenie cez port
Parametre:
port - na ktorom sa očakáva pripojenie
Výnimky:
java.io.IOException - v prípade keď port je obsadený (alebo inej sieťovej chyby)

Metódy

public void **die**()
Metóda pre čisté pozastavenie vlákna

public static void **processIPFIXData**(java.net.Inet4Address ipmb,
IPFIXTemplateRecord ipfixTemplate,
IPFIXDataRecord ipfixData)
Predá Ip adresu meracieho bodu, IPFIX šablónu a údaje pre jednotlivé vlákna

Parametre:

ipmb - IP adresa meracieho bodu

ipfixTemplate - IPFIX šablóna

ipfixData - IPFIX údaj

public void **run**()

Vytvorí Workers a čaká na pripojenie

Špecifikované:

run in interface java.lang.Runnable

Overrides:

run in class java.lang.Thread

4.2.2 Trieda ACPIPFIXWorker

Trieda, ktorá umožňuje filtrovanie údajov, ich následné posielanie protokolom ACP, a komunikáciu Analyzérom.

Konštruktory

public **ACPIPFIXWorke**r()

Null konštruktor pre toto vlákno

public **ACPIPFIXWorker**(java.lang.ThreadGroup group,
int i)

Konštruktor, nastaví meno a id pre tento worker.

Parametre:

group - ThreadGroup ku ktorému worker patrí

i - int id no.

Metódy

public void **die**()

Metóda pre čisté zastavenie Threadu

public static void **processRequest**(java.net.Socket request)

Spracuje spojenie z pool

Parametre:

request - Socket na ktorom prišlo spojenie

public boolean **ACPauthentication**(java.net.Socket connection,
java.lang.String login,
java.lang.String passwd)
throws java.io.IOException

Metóda, ktorá slúži na overenie autentifikovaného spojenia.

Parametre:

connection - spojenie, cez ktoré sa očakávajú autentifikačné údaje

login - prijatý login

passwd - prijaté heslo

Návratová hodnot:

true - ak sa došlo k správnej autentifikácie, false - ak analyzer poslal nesprávne autentifikačné údaje.

Výnimky:

java.io.IOException

public void **ACPCommunication**(int c)
throws java.io.IOException

Metóda, ktorá slúži na prijímanie a následné reagovanie správ od analyzera.

Výnimky:

java.io.IOException

public int **stringToInt**(java.lang.String string)

Metóda na prekonvertovanie Stringu na Int.

Parametre:

string - String, ktorý sa má prekonvertovať na Int

Návratová hodnot:

prekonvertovaný String na Int

public void **dispatchIPFIXRecord**(java.net.Inet4Address ipmb,
IPFIXTemplateRecord template,
IPFIXDataRecord data)

Metéda, ktorá sprostredkuje IPFIX údaje na spracovanie.

Parametre:

ipmb - IP adresa meracieho bodu

template - IPFIX šablóna

data - IPFIX údaje

public void **run**()

hlavný loop pre ACP, kde sa kontroluje spojenie, prebieha príjem riadiacich správ a samotné posielanie dát vyhovujúce šablóne a filtračným kritériam.

Špecifikované:

run in interface java.lang.Runnable

Overrides:

run in class java.lang.Thread

4.2.3 Trieda ACPIPFIXTemplate

Trieda balíku sk.tuke.cnl.bm obsahuje identifikátory informačných elementov, ktoré sú v súčasnom stave Exportéra podporované. Zoznam týchto informačných elementov sú znázornené v tabuľke 4 – 1.

Tabuľka 4 – 1 Informačné elementy podporované ACPapim

Názov	Identifikátor	Enterprise id
protocolIdentifier	4	-
sourceTransportPort	7	-
sourceIPv4Address	8	-
destinationTransportPort	11	-
destinationIPv4Address	12	-
packetTotalCount	86	-
observationPointID	138	-
flowStartNanoseconds	156	-
flowEndNanoseconds	157	-
icmpTypeIPv4	176	-
icmpCodeIPv4	177	-
tcpSynTotalCount	218	-
tcpFinTotalCount	219	-
tcpRstTotalCount	220	-
tcpPshTotalCount	221	-
tcpAckTotalCount	222	-
tcpUrgTotalCount	223	-
roundTripTimeNanoseconds	240	26235
int packetPairsTotalCount	241	26235

4.2.4 Trieda Filter

Trieda obsahujúca súbor filtrovacích pravidiel premávky na základe IP adresy meracieho bodu, zdrojovej a cieľovej IP adresy, skupiny portov (zdrojových aj cieľových) a protokolov. Poskytuje metódy na overenie či sa určitá IP adresa, port alebo protokol vyhovuje nastaveným filtrovacím kritériám.

Atribúty

protected java.util.ArrayList<java.nio.ByteBuffer[]>**mpIP**

protected java.util.ArrayList<java.nio.ByteBuffer[]>**srcIP**

protected java.util.ArrayList<java.nio.ByteBuffer[]>**dstIP**

protected java.util.ArrayList<java.lang.String>**mpIPStr**

protected java.util.ArrayList<java.lang.String>**srcIPStr**

protected java.util.ArrayList<java.lang.String>**dstIPStr**

protected java.util.ArrayList<int[]>**srcPorts**

protected java.util.ArrayList<int[]>**dstPorts**

protected java.util.ArrayList<int[]>**protocols**

Konštruktor

public **Filter**()

Vytvorí nový filter bez pravidiel.

Metódy

public void **addMP**(java.lang.String ipStr)

Pridá IP adresu a sieťovú masku meracieho bodu.

Parametre:

ipStr - Reťazec reprezentujúci IP adresu meracieho bodu (formát viď. InetAddr.parse)

Výnimky:

java.lang.NullPointerException - Viď. InetAddr.parse

java.text.ParseException - Viď. InetAddr.parse

public void **addMP**(byte[][] ip)

Pridá IP adresu a sieťovú masku meracieho bodu.

Parametre:

ip - IP adresa a sieťová maska (formát reťazca viď. InetAddr.verify)

Výnimky:

java.lang.NullPointerException - InetAddr.verify

java.lang.NumberFormatException - InetAddr.verify

public void **addSrcIP**(java.lang.String ipStr)

Pridá zdrojovú IP adresu a sieťovú masku.

Parametre:

ipStr - Reťazec reprezentujúci IP adresu (formát viď. InetAddr.verify)

Výnimky:

java.lang.NullPointerException - Viď. InetAddr.parse

java.text.ParseException - Viď. InetAddr.parse

public void **addSrcIP**(byte[][] ip)

Pridá zdrojovú IP adresu a sieťovú masku.

Parametre:

ip - IP adresa a sieťová maska (formát viď. InetAddr.verify)

Výnimky:

java.lang.NumberFormatException - InetAddr.verify

public void **addSrcPorts**(java.lang.String portsStr, java.lang.String sep,
java.lang.String rangeSep)

Pridá zdrojové porty.

Parametre:

portsStr - Reťazec obsahujúci zoznam zdrojových portov (formát viď.
InetAddr.parseIntervalsString)

sep - Viď. InetAddr.parseIntervalsString

rangeSep - Viď. InetAddr.parseIntervalsString

Výnimky:

java.text.ParseException - Viď. InetAddr.parseIntervalsString

java.lang.NumberFormatException - Viď. InetAddr.parseIntervalsString

public void **addDstIP**(java.lang.String ipStr)

Pridá cieľovú IP adresu a sieťovú masku.

Parametre:

ipStr - Reťazec reprezentujúci IP adresu (formát viď. InetAddr.verify)

Výnimky:

java.lang.NullPointerException, java.text.ParseException - Viď. InetAddr.parse

public void **addDstIP**(byte[][] ip)

Pridá cieľovú IP adresu a sieťovú masku.

Parametre:

ip - IP adresa a sieťová maska (formát viď. InetAddr.verify)

Výnimky:

java.lang.NumberFormatException - InetAddr.verify

public void **addDstPorts**(java.lang.String portsStr, java.lang.String sep,
java.lang.String rangeSep)

Pridá cieľové porty.

Parametre:

portsStr - Reťazec obsahujúci zoznam cieľových portov (formát viď. InetAddr.parseIntervalsString)

sep - Viď. InetAddr.parseIntervalsString

rangeSep - Viď. InetAddr.parseIntervalsString

Výnimky:

java.text.ParseException - Viď. InetAddr.parseIntervalsString

java.lang.NumberFormatException - Viď. InetAddr.parseIntervalsString

public void **addProtocols**(java.lang.String protStr, java.lang.String sep,
java.lang.String rangeSep)

Pridá protokoly.

Parametre:

protStr - Reťazec obsahujúci zoznam protokolov (formát viď.
InetAddr.parseIntervalsString)

sep - Viď. InetAddr.parseIntervalsString

rangeSep - Viď. InetAddr.parseIntervalsString

Výnimky:

java.text.ParseException - Viď. InetAddr.parseIntervalsString

java.lang.NumberFormatException - Viď. InetAddr.parseIntervalsString

public void **mergeFilter**(Filter filter)

Pripojí zadaný filter k existujúcemu a vykoná optimalizačné kroky: - spojenie a agregáciu pravidiel obsahujúcich IP adresy (IP adresy meracích bodov, zdrojové a cieľové IP adresy), - spojenie a združenie pravidiel obsahujúcich rozsahy (zdrojové a cieľové porty, protokoly). Pravidlá sú spájané a ak je niektoré pravidlo podmnožinou iného, je pohltené. Ak napríklad vo filtri neexistuje filtračné kritérium pre zdrojový port, a pripájané pravidlo (to, ktoré je predané cez parameter metódy) obsahuje filtračné kritérium pre zdrojový port 80, výsledné filtračné kritérium pre zdrojový port po spojení ostane prázdne, (čo znamená žiadny filter a teda všetky porty), a každé takéto spojenie pre zdrojový port dopadne rovnako, pretože všetky užšie kritériá sú podmnožinou tohto kritéria.

Parametre:

filter - filter obsahujúci filtračné kritériá ktoré sa spoja s existujúcimi

public java.util.ArrayList<java.nio.ByteBuffer[]> **getMP**()

Vráti IP adresu a sieťovú masku meracieho bodu.

Návratová hodnota:

IP adresa a sieťová maska meracieho bodu ako ByteBuffer[2], kde [0] obsahuje ByteBuffer nad poľom byte[4] pre IP adresu a [1] pre sieťovú masku

public java.lang.String **getMPString**(java.lang.String sep)

Vráti IP adresu a sieťovú masku meracieho bodu ako reťazec.

Parametre:

sep - Reťazec oddeľujúci jednotlivé IP adresy

Návratová hodnota:

IP adresa a sieťová maska meracieho bodu (formát viď. InetAddr.toString)

public java.util.ArrayList<java.nio.ByteBuffer[]> **getSrcIP**()

Vráti zdrojovú IP adresu a sieťovú masku.

Návratová hodnota:

Zdrojová IP adresa a sieťová maska ako ByteBuffer[2], kde [0] obsahuje ByteBuffer nad poľom byte[4] pre IP adresu a [1] pre sieťovú masku

public java.lang.String **getSrcIPString**(java.lang.String sep)

Vráti zdrojovú IP adresu a sieťovú masku ako reťazec.

Parametre:

sep - Reťazec oddeľujúci jednotlivé IP adresy

Návratová hodnota:

Zdrojová IP adresa a sieťová maska (formát viď. InetAddr.toString)

public java.util.ArrayList<int[]> **getSrcPorts**()

Vráti pole so zdrojovými portami.

Návratová hodnota:

Pole so zdrojovými portami (formát viď. návratová hodnota InetAddr.parseIntervalsString)

public java.lang.String **getSrcPortsString**(java.lang.String sep,
java.lang.String rangeSep)

Vráti zoznam zdrojových portov ako reťazec.

Parametre:

sep - Viď. InetAddr.intervalsToString

rangeSep - Viď. InetAddr.intervalsToString

Návratová hodnota:

Viď. InetAddr.intervalsToString

public java.util.ArrayList<java.nio.ByteBuffer[]> **getDstIP**()

Vráti cieľovú IP adresu a sieťovú masku.

Návratová hodnota:

Cieľová IP adresa a sieťová maska ako ByteBuffer[2], kde [0] obsahuje ByteBuffer nad poľom byte[4] pre IP adresu a [1] pre sieťovú masku

public java.lang.String **getDstIPString**(java.lang.String sep)

Vráti cieľovú IP adresu a sieťovú masku ako reťazec.

Parametre:

sep - Reťazec oddeľujúci jednotlivé IP adresy

Návratová hodnota:

Cieľová IP adresa a sieťová maska (formát viď. InetAddr.toString)

public java.util.ArrayList<int[]> **getDstPorts**()

Vráti pole s cieľovými portami.

Návratová hodnota:

Pole s cieľovými portami (formát viď. návratová hodnota InetAddr.parseIntervalsString)

public java.lang.String **getDstPortsString**(java.lang.String sep,
java.lang.String rangeSep)

Vráti zoznam cieľových portov ako reťazec.

Parametre:

sep - Viď. InetAddr.intervalsToString

rangeSep - Viď. InetAddr.intervalsToString

Návratová hodnota:

Viď. InetAddr.intervalsToString

public java.util.ArrayList<int[]> **getProtocols**()

Vráti pole s protokolmi.

Návratová hodnota:

Pole s protokolmi (formát viď. návratová hodnota InetAddr.parseIntervalsString)

public java.lang.String **getProtocolsString**(java.lang.String sep,
java.lang.String rangeSep)

Vráti zoznam protokolov ako reťazec.

Parametre:

sep - Viď. InetAddr.intervalsToString

rangeSep - Viď. InetAddr.intervalsToString

Návratová hodnota:

Viď. InetAddr.intervalsToString

public void **aggregateMPRules**()

Agreguje filtračné pravidlá meracích bodov.

public void **aggregateSrcIPRules**()

Agreguje filtračné pravidlá zdrojových IP adries.

public void **aggregateDstIPRules**()

Agreguje filtračné pravidlá cieľových IP adries.

public SimpleFilter **createSimpleFilter**()

Podľa nastavených filtračných kritérií, vytvorí SimpleFilter.

Návratová hodnota:

inštanciu objektu SimpleFilter

Výnimky:

InvalidFilterRuleException - viď SimpleFilter

public void **ACPCreateFilter**(java.lang.String MpIP,
java.lang.String SrcIP,
java.lang.String SrcPort,
java.lang.String DstIP,
java.lang.String DstPort,
java.lang.String Protocol)

Vytvor filter podľa zvolených filtračných kritérií pre ACPapi

Parametre:

MpIP - IP Adresa meracieho bodu

SrcIP - IP Adresa zdroja

SrcPort - Číslo portu zdroja

DstIP - Adresa cieľa

DstPort - Číslo portu cieľa

Protocol - Číslo protokolu

4.2.5 Trieda InetAddr

Poskytuje funkcie na prácu s IPv4 adresami a portami.

Konštruktor

public **InetAddr**()

Metódy

public static void **verify**(byte[][] addr)
Overí platnosť IP adresy a jej sieťovej masky (overuje len formu, nie obsah).
Parametre:
addr - [0][4] - IP adresa (v jednotlivých bytoch poľa sú uložené príslušné oktety IP adresy), [1][4] - Sieťová maska (v jednotlivých prvkoch poľa sú uložené príslušné oktety sieťovej masky).
Výnimky:
java.lang.NullPointerException - Ak je hodnota parametra alebo poľa s IP adresou resp. sieťovou maskou rovná null
java.lang.NumberFormatException - Ak parameter obsahuje viac ako dve polia (jedno pre IP adresu, druhé pre sieťovú masku), alebo dĺžka jedného z polí sa nerovná štyrom

public static boolean **equals**(byte[][] addr1, byte[][] addr2)
Porovná zhodnosť dvoch IP adries.
Parametre:
addr1 - Prvá IP adresa (Vstupný parameter verify)
addr2 - Druhá IP adresa (Vstupný parameter verify)
Návratová hodnota:
true ak sa rovnajú adresy aj ich sieťové masky, false ak sa adresy líšia
Výnimky:
java.lang.NullPointerException - verify

java.lang.NumberFormatException - Ak je neplatná jedna z adries, alebo sa nezhodujú sieťové masky

public static byte[][] **parse**(java.lang.String addr)
Prevedie reťazec s IP adresou a maskou na pole bytov.
Parametre:
addr - Reťazec v tvare A.B.C.D/n, kde A.B.C.D je IP adresa a n je počet bitov sieťovej časti adresy. Časť /n je nepovinná a v prípade, že nebude uvedená, bude za n dosadená hodnota 32.
Návratová hodnota:
[0][4] - pole obsahujúce IP adresu, [1][4] - pole obsahujúce sieťovú masku
Výnimky:
java.lang.NumberFormatException - Ak je hodnota niektorého z oktetov IP adresy mimo intervalu [0,255]
java.text.ParseException - Ak vstupný reťazec nemá požadovaný formát
java.lang.NullPointerException - Ak je vstupný reťazec rovný null

public static byte[][] **convert**(long addr, long mask)
Prevedie adresu a sieťovú masku na IP adresu vo formáte byte[2][4].
Parametre:
addr - IP adresa
mask - Sieťová maska
Návratová hodnota:
[0][4] - IP adresa (v jednotlivých bytoch poľa sú uložené príslušné oktety IP adresy), [1][4] - Sieťová maska (v jednotlivých bytoch poľa sú uložené príslušné oktety sieťovej masky).
Výnimky:
java.lang.NumberFormatException - Ak je IP adresa alebo sieťová maska mimo intervalu [-2147483648,2147483647], teda mimo intervalu ktorý zahŕňa typ int

public static java.lang.String **toString**(byte[][] addr)

Vráti IP adresu a masku ako reťazec.

Parametre:

addr - IP adresa (Vstupný parameter verify)

Návratová hodnota:

IP adresu vo formáte A.B.C.D/n, kde A.B.C.D je IP adresa a n je počet bitov sieťovej časti IP adresy

Výnimky:

java.lang.NullPointerException - verify

java.lang.NumberFormatException - verify

public static java.lang.String **toString**(int addr, int mask)

Vráti IP adresu a masku ako reťazec.

Parametre:

addr - IP adresa

mask - Sieťová maska

Návratová hodnota:

IP adresu vo formáte A.B.C.D/n, kde A.B.C.D je IP adresa a n je počet bitov sieťovej časti IP adresy

public static byte[] **netbitsToMask**(int n)

Konvertuje číslo označujúce počet bitov sieťovej časti IP adresy na sieťovú masku.

Parametre:

n - Počet bitov sieťovej časti IP adresy

Návratová hodnota:

Sieťovú masku, kde každý prvok poľa zodpovedá príslušnému oktetu sieťovej masky

Výnimky:

java.lang.NumberFormatException - V prípade, že n je mimo intevalu [0,32]

public static byte[] **netbitsToMask**(java.lang.String nStr)

Konvertuje číslo označujúce počet bitov sieťovej časti IP adresy na sieťovú masku.

Parametre:

nStr - Počet bitov sieťovej časti IP adresy

Návratová hodnota:

Sieťovú masku, kde každý prvok poľa zodpovedá príslušnému oktetu sieťovej masky

Výnimky:

java.lang.NumberFormatException - V prípade, že vstupný reťazec nie je platné číslo int, alebo n je mimo intevalu [0,32]

public static int **maskToNetbits**(byte[] mask)

Konvertuje sieťovú masku na počet bitov sieťovej časti IP adresy.

Parametre:

mask - Sieťová maska (v jednotlivých prvkoch poľa sú uložené príslušné oktety sieťovej masky)

Návratová hodnota:

Počet bitov sieťovej časti IP adresy

Výnimky:

java.lang.NumberFormatException - Ak dĺžka vtupného poľa sa nerovná 4, alebo ak je sieťová maska v neplatnom fomáte (binárna sekvencia 1 nasledovaná sekvenciou 0)

public static int **maskToNetbits**(int mask)

Konvertuje sieťovú masku na počet bitov sieťovej časti IP adresy.

Parametre:

mask - Sieťová maska

Návratová hodnota:

Počet bitov sieťovej časti IP adresy

public static int **compareMasks**(int mask1, int mask2)

Porovná dve sieťové masky a zistí ktorá je väčšia.

Parametre:

mask1 - prvá sieťová maska

mask2 - druhá sieťová maska

Návratová hodnota:

-1 ak má prvá maska menej bitov ako druhá, 0 ak sú zhodné, 1 ak má prvá maska viac bitov ako druhá

Netestuje sa či má maska správny formát, teda či sa jedná o sekvenciu jednotiek nasledovanú sekvenciou núl. V prípade, že má maska nesprávny formát, funkcia síce vráti hodnotu -1, 0, alebo 1, tá však nemá zmysel, pretože neplatné masky nemá zmysel porovnávať.

public static int[][] **parseIntervalsString**(java.lang.String inStr, java.lang.String sep, java.lang.String rangeSep, int minVal, int maxVal)

Prevedie reťazec s číslami portov na pole s ich hodnotami.

Parametre:

inStr - Reťazec s číslami portov. Jednotlivé čísla a rozsahy sú oddelené reťazcom sep a hranice rozsahov sú oddelené reťazcom rangeSep. Ak platí, že sep = „,“ a rangeSep = „-“, potom platný formát vstupného reťazca je napr. a,b-c,d-e,f,...

sep - Reťazec oddeľujúci jednotlivé porty a intervaly

rangeSep - Reťazec oddeľujúci hranice rozsahov

minVal - Minimálna povolená hodnota

maxVal - Maximálna povolená hodnota

Návratová hodnota:

Pole dvojprvkových polí s rozsahmi portov, kde v prvom prvku je spodná a v druhom horná hranica rozsahu (ak sa nejedná o interval, obe hodnoty sú rovnaké).

Napr. reťazec 20-23,25,80 vráti pole int[3][2] s hodnotami prvkov:

[0][0]=20, [0][1]=23, [1][0]=25, [1][1]=25, [2][0]=80, [2][1]=80

Výnimky:

java.text.ParseException - Ak vstupný parameter nemá požadovaný formát

java.lang.NumberFormatException - Ak je hodnota jedného z portov mimo intervalu [minVal,maxVal]

public static java.lang.String **intervalsToString**(java.util.ArrayList<int[]> inList, java.lang.String sep, java.lang.String rangeSep)

Vráti zoznam portov ako reťazec.

Parametre:

inList - Porty a rozsahy, ktoré sa majú previesť na reťazec

sep - Reťazec oddeľujúci jednotlivé porty a intervaly

rangeSep - Reťazec oddeľujúci hranice rozsahov

Návratová hodnota:

Reťazec reprezentujúci textovú formu zoznamu portov, kde jednotlivé porty a intervaly sú oddelené reťazcom sep a hranice rozsahov sú oddelené reťazcom rangeSep

public static java.lang.String **intervalsToString**(int[][] inList, java.lang.String sep, java.lang.String rangeSep)

Vráti zoznam portov ako reťazec.

Parametre:

inList - Porty a rozsahy, ktoré sa majú previesť na reťazec

sep - Reťazec oddeľujúci jednotlivé porty a intervaly

rangeSep - Reťazec oddeľujúci hranice rozsahov

Návratová hodnota:

Reťazec reprezentujúci textovú formu zoznamu portov, kde jednotlivé porty a intervaly sú oddelené reťazcom sep a hranice rozsahov sú oddelené reťazcom rangeSep

public static boolean **contains**(int addr1, int mask1, int addr2, int mask2)

Testuje, či prvá IP adresa patrí do siete, ktorú definuje druhá IP adresa.

Parametre:

addr1 - Prvá IP adresa

mask1 - Maska prvej IP adresy

addr2 - Druhá IP adresa

mask2 - Maska druhej IP adresy

Návratová hodnota:

true prvá IP adresa patrí do siete, ktorú definuje druhá IP adresa alebo je s ňou zhodná, inak false

4.2.6 Trieda InvalidFilterRuleException

Výnimka - vzniká, ak pravidlo filtra nemá požadovaný formát.

4.2.7 Trieda SimpleFilter

Trieda na zoskupenie týchto filtrovacích pravidiel: IP adresy meracích bodov, zdrojové a cieľové IP adresy, zoznam zdrojových a cieľových portov, zoznam protokolov. Poskytuje metódy na overenie či určitá IP adresa, port alebo protokol vyhovuje týmto filtračným pravidlám.

Konštruktor

public **SimpleFilter**(int[][] mpIP, int[][] srcIP, int[][] dstIP, int[][] srcPorts,
int[][] dstPorts, int[][] protocols)

Vytvorí nový filter so zadanými filtračnými pravidlami.

Formát mpIP, srcIP a dstIP pre i-té pravidlo:

int[i][0] - IP adresa siete (resp. hosta, ak je maska 255.255.255.255)

int[i][1] - Sieťová maska

Formát srcPorts, dstPorts a protocols pre i-té pravidlo:

int[i][0] - Spodná hranica intervalu

int[i][1] - Horná hranica intervalu

Parametre:

mpIP - Pole so sieťovými adresami a maskami meracích bodov

srcIP - Pole so zdrojovými sieťovými adresami a maskami

dstIP - Pole s cieľovými sieťovými adresami a maskami

srcPorts - Pole so zdrojovými portami

dstPorts - Pole s cieľovými portami

protocols - Pole s protokolmi

Výnimky:

InvalidFilterRuleException - Ak niektorý zo vstupných parametrov nemá požadovaný formát

Metódy

public int **getFlag**()

Vráti flag súhrnný flag pre jednotlivé pravidlá filtra.

Návratová hodnota:

flag daného filtra

public boolean **mpMatches**(byte[] ip)

Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre merací bod. V prípade, že je vo filtri sieťovou maskou nastavená celá sieť, zistí, či adresa patrí do tejto siete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

ip - IP adresa meracieho bodu vo formáte byte[4], kde každý prvok poľa zodpovedá príslušnému oktetu IP adresy

Návratová hodnota:

false ak sa IP adresa nezhoduje s IP adresou meracieho bodu, resp. ak nepatrí do definovanej siete, inak vráti true

Výnimky:

java.lang.NullPointerException - Ak má vstupný parameter hodnotu null

java.lang.NumberFormatException - Ak má vstupný parameter neplatný počet oktetov (iný ako 4)

public boolean **mpMatches**(int ip)

Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre merací bod. V prípade, že je vo filtri sieťovou maskou nastavená celá sieť, zistí, či adresa patrí do tejto siete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

ip - IP adresa meracieho bodu

Návratová hodnota:

false ak sa IP adresa nezhoduje s IP adresou meracieho bodu, resp. ak nepatrí do definovanej siete, inak vráti true

public boolean **srcIPMatches**(byte[] ip)

Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre zdrojovú IP adresu. V prípade, že je vo filtri sieťovou maskou nastavená celá (pod)sieť, zistí, či adresa patrí do tejto podsiete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

ip - Zdrojová IP adresa vo formáte byte[4], kde každý prvok poľa zodpovedá príslušnému oktetu IP adresy

Návratová hodnota:

false ak sa IP adresa nezhoduje so zdrojovou IP adresou, resp. ak nepatrí do definovanej siete, inak vráti true

Výnimky:

java.lang.NullPointerException - Ak má vstupný parameter hodnotu null

java.lang.NumberFormatException - Ak má vstupný parameter neplatný počet oktetov (iný ako 4)

public boolean **srcIPMatches**(int ip)

Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre zdrojovú IP adresu. V prípade, že je vo filtri sieťovou maskou nastavená celá (pod)sieť, zistí, či adresa patrí do tejto podsiete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

ip - Zdrojová IP adresa

Návratová hodnota:

false ak sa IP adresa nezhoduje so zdrojovou IP adresou, resp. ak nepatrí do definovanej siete, inak vráti true

public boolean **srcPortMatches**(int port)
Overí, či sa daný port nachádza v zozname zdrojových portov. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.
Parametre:
port - Hodnota zdrojového portu
Návratová hodnota:
false ak sa port nenachádza v zozname zdrojových portov, inak vráti true

public boolean **dstIPMatches**(byte[] ip)
Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre cieľovú IP adresu. V prípade, že je vo filtri sieťovou maskou nastavená celá (pod)sieť, zistí, či adresa patrí do tejto podsiete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.
Parametre:
ip - Cieľová IP adresa vo formáte byte[4], kde každý prvok poľa zodpovedá príslušnému oktetu IP adresy
Návratová hodnota:
false ak sa IP adresa nezhoduje s cieľovou IP adresou, resp. ak nepatrí do definovanej siete, inak vráti true
Výnimky:
java.lang.NullPointerException - Ak má vstupný parameter hodnotu null
java.lang.NumberFormatException - Ak má vstupný parameter neplatný počet oktetov (iný ako 4)

public boolean **dstIPMatches**(int ip)

Overí, či daná IP adresa vyhovuje nastaveným kritériám filtra pre cieľovú IP adresu. V prípade, že je vo filtri sieťovou maskou nastavená celá (pod)sieť, zistí, či adresa patrí do tejto podsiete, inak len overí, či sa IP adresy zhodujú. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

ip - Cieľová IP adresa

Návratová hodnota:

false ak sa IP adresa nezhoduje s cieľovou IP adresou, resp. ak nepatrí do definovanej siete, inak vráti true

public boolean **dstPortMatches**(int port)

Overí, či sa daný port nachádza v zozname zdrojových portov. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

port - Hodnota cieľového portu

Návratová hodnota:

false ak sa port nenachádza v zozname cieľových portov, inak vráti true

public boolean **protocolMatches**(int protocol)

Overí, či sa daný protokol nachádza v zozname protokolov. Ak nie je vo filtri nastavená žiadna hodnota, považuje sa kritérium za splnené.

Parametre:

protocol - Hodnota protokolu

Návratová hodnota:

false ak sa protokol nenachádza v zozname protokolov, inak vráti true

public java.lang.String **toString**()

Na získanie textovej reprezentácie pravidiel nastavených vo filtri.

Návratová hodnota:

Všetky pravidlá filtra vo forme reťazca.

5 Preklad programu

5.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe bakalárskej práce.

Sú k dispozícii tieto zdrojové texty:

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

```
- balíček sk.tuke.cnl.bm.JXColl.accounting:
   AccountingManager.java
   AccountingRecord.java
   AccountingRecordsCache.java
   AccountingRecordsExporter.java
- balíček sk.tuke.cnl.bm.JXColl.IPFIX:
   FieldSpecifier.java
   IPFIXDataRecord.java
   IPFIXMessage.java
   IPFIXOptionsTemplateRecord.java
   IPFIXSet.java
   IPFIXTemplateRecord.java
   IPFIXTemplateCache.java
```

5.2 Požiadavky na technické prostriedky pri preklade

Preklad programu si vyžaduje nasledovnú hardvérovú konfigurácia:

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

5.3 Požiadavky na programové prostriedky pri preklade

- ľubovoľný operačný systém s podporou Java Virtual Machine (JVM) (Windows XP/Server 2003/Vista, Linux alebo Solaris)
- Java Runtime Environment (JRE) verzie 1.6.0 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 strednej vrstvy architektúry BasicMeter. Z toho vyplýva jeho náväznosť na modul, ktorý pracuje na prvej vrstve spomínanej architektúry, teda na Exportér.

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 obľubený java IDE, kde stačí jednoducho nastaviť verziu JDK na 5.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam.

6 Zhodnotenie riešenia

Hlavným cieľom práce bolo vytvorenie podpory priameho pripojenia prostredníctvom protokolu ACP, a to rozšírením existujúceho riešenia programu JXColl. Keďže predošlá verzia neobsahovala niektoré prvky, ktoré boli potrebné pre spracovanie a nachystanie IPFIX údajov pre priamy prenos, bolo potrebné dopracovať chýbajúce časti architektúry.

Súčasná verzia programu dovoľuje zachytávanie a ukladanie IPFIX údajov do databázy alebo ich sprístupniť cez protokol ACP, a zároveň podporuje generovanie záznamov, potrebné pre účtovaciu aplikáciu.

Riešenie modulu pre priame pripojenie bolo testované na reálnej prevádzke, a fungovalo spoľahlivo (viď bakalárska práca Vladimíra Závadu 2009). Modul pri viacerých súbežných prepojeniach nebol dostatočne dlho testovaný kvôli časovej náročnosti takého procesu, a preto pri jeho použití sa môžu objaviť nepredvídané chyby.

Predošlá verzia programu čelila niekoľkým chybám, ktoré počas vývoja boli čiastočne opravené. Odstránenie najzávažnejšej chyby, ktorá sa prejavovala zaplnením pamäte a

následným pádom celého programu, je nad rámec tejto práce, a preto jej odstránenie sa prenechá pre budúci vývoj meracej platformy BasicMeter.

7 Zoznam použitej literatúry

[1] Koščo, M.: Opis sieťových protokolov prostredníctvom jazyka XML. Diplomová práca (vedúci Ing. J. Genči), Košice: KPI FEI TU, 2005

[2] Kaščák, M.: Príspevok k problematike aplikačného využitia meraní prevádzkových parametrov počítačových sietí Diplomová práce (vedúci Ing. F. Jakab, PhD.), Košice: KPI FEI TU, 2007