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

Optimalizácia zhromažďovacieho procesu nástroja BasicMeter

Diplomová práca

Príloha A

SYSTÉMOVÁ PRÍRUČKA JXColl v3.6

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Juraj Giertl, PhD.
Konzultant:	Ing. Martin Révés, PhD.

Košice 2011 **Bc. Adrián Pekár**

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.

Jednosmerné oneskorenie predstavuje čas, za ktorý sa dostane paket z jedného bodu do druhého. Aby hodnota oneskorenia bola správna, je potrebné aby hodiny oboch koncových bodov boli zosynchronizované.

Nová verzia JXColl bola rozšírená o dva nové moduly určené pre meranie jednosmerného oneskorenia medzi dvoma meracími bodmi. Synchronizačný modul predstavuje synchronizačný server, ktorý na každý prijatý paket odpovie svojím lokálnym časom. Modul pre meranie jednosmerného oneskorenia predspracováva údajov, na základe ktorého určuje hodnoty owd a ukladá ich do databázy.

Program bol vytvorený Ľubošom Koščom, neskôr zoptimalizovaný a doplnený novými funkciami Michalom Kaščákom, Adriánom Pekárom a Tomášom Vereščákom.

2 Analýza riešenia

Jednosmerné oneskorenie sa meria pomocou prenosu časovej známky medzi dvoma zosynchronizovanými koncovými bodmi. Ako koncové body v architektúre mera-

cieho nástroja boli zvolené exportovacie procesy. Z IPFIX informačných elementov boli pre účely prenesenia časových známok vybrané informačné elementy zo skupiny času začiatku (*flowStart*) a ukončenia (*flowEnd*) toku. Pomocou navrhnutých informačných elementov (*firstPacketID* a *lastPacketID*) sa umožnila identifikácia rovnakých prvých alebo posledných IP paketov zachytených v dvoch rôznych meracích bodoch.

Synchronizácia hodín meracích bodov bola implementovaná iba z časti. Exportéry posielajú pakety slúžiace na synchronizáciu voči kolektoru, ich obsluha kolektorom však ešte nebola vytvorená. Ani samotný výpočet jednosmerného oneskorenia nebol zabezpečený kolektorom.

Modul pre meranie jednosmerného oneskorenia bol implementovaný do už existujúcej architektúry zhromažďovacieho procesu meracieho nástroja BasicMeter. Keďže sa jedná o rozšírenie existujúceho riešenia, prostriedky vývoja sa nemenili. Koncepcia a štruktúra kolektora sa zachovala, pričom pre účely synchronizácie meracích bodov, v kolektore bolo vytvorené samostatné a na funkčných častiach nezávislé vlákno. Pre výpočet jednosmerného oneskorenia bol v kolektore vytvorený balík tried s názvom *sk.tuke.cnl.bm.JXColl.OWD*.

2.1 Synchronizácia hodín meracích bodov

Synchronizačné vlákno počúva na vopred dohodnutom ale zároveň ľubovoľne nastaviteľnom porte, ktorý po obdržaní každého paketu s časovou známkou vykoná nasledujúce kroky:

- zistí aktuálny čas prijatia danej časovej známky,
- vykoná nad zisteným časom úpravu,
- upravený čas následne zapíše do prijatej časovej známky na pozíciu určenú exportérom,

- pošle paket naspäť na tú adresu, od ktorej danú časovú známku obdržal.

2.2 Výpočet jednosmerného oneskorenia

Metóda výpočtu jednosmerného oneskorenia prestavuje akési predspracovávanie prijatých informácií o tokoch, ktorá bola zaradená pred pôvodný proces spracovania údajov. Kolektor po prijatí každej IPFIX správy zistí, či vyhovuje filtračným kritériám prvého a druhého stupňa. Následne údaje potrebné pre výpočet jednosmerného oneskorenia uloží do vzájomne si konkurujúcich zásobníkov. V ďalšom kroku, porovnaním údajov v zásobníkoch hľadá zodpovedajúce časové známky. Časové známky sa považujú za zodpovedajúce, keď spĺňajú nasledujúce podmienky:

- záznamy o toku prišli z rôznych meracích bodov,
- obe sú absolútne časy začiatku (*flowStartNanoseconds*) alebo ukončenia (*flowEndNanoseconds*) toku,
- obe majú zhodné prislúchajúce identifikátory prvého (*firstPacketID*) a posledného (*lastPacketID*) IP paketu,
- obe patria do toho istého toku.

V prípade, že všetky podmienky sú splnené, kolektor vypočíta hodnotu jednosmerného oneskorenia a uloží do navrhnutej tabuľky. Následne, dátové záznamy z oboch zásobníkov, na základe ktorých vypočítal jednosmerné oneskorenie, presunie pre pôvodné spracovanie.

V kolektore bolo vytvorené ďalšie vlákno, ktoré v každej sekunde prehľadáva zásobníky a podľa časovej známky, ktorá je zaznamenaná pri každom zápise, určuje, či daný dátový záznam už prekročil maximálnu dobu čakania alebo nie. V prípade, že doba čakania je prekročená, vlákno skupinu hodnôt zo zásobníka vymaže a dátový záznam, ktorý k nej patrí, pošle pre pôvodné spracovanie.

3 Popis riešenia

Jednotlivé časti programu JXColl sú umiestnené v nasledujúcich balíkoch:

- sk.tuke.cnl.bm − spoločné triedy pre celý projekt
- sk.tuke.cnl.bm.JXColl − triedy samotného programu
- sk.tuke.cnl.bm.JXColl.IPFIX − triedy s manuálnou implementáciou IPFIX
- sk.tuke.cnl.bm.JXColl.OWD − triedy určené na predspracovávanie údajov a výpočet jednosmerného oneskorenia
- sk.tuke.cnl.bm.JXColl.accounting − triedy určené na export údajov pre účtovaciu aplikáciu
- sk.tuke.cnl.bm.JXColl.export − triedy určené na export dát do databázy alebo protokolom ACP

Jednotlivé časti modulu jednosmerného oneskorenia sú implementované v balíku OWD. Popis týchto tried, spolu s jej konštruktormi, metódami a výnimkami bude uvedený v nasledujúcej časti príručky.

3.1 Popis tried, členských premenných, metód a výnimiek

Kedže sa počas vývoja pôvodné triedy a ich metódy nezmenili, v nasledujúcich častiach budú uvedené len tie, ktoré sú z implementačného pohľadu modulu pre jednosmerné oneskorenie potrebné. Popis ostatných tried a metód je uvedený v príručkách predošlých verzií programu.

3.1.1 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.

Hádže:

InterruptedException – keď je prerušený počas čakania na zápis do zásobníka.

NullPointerException – keď vstup je null hodnota.

SQLException – keď je chyba počas prístupu na databázu.

```
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 B. 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 pektu.

addr − InetAddress adresa od ktorej paket prišiel.

Hádže:

InterruptedException − keď je prerušený počas čakania na zápis do zásobníka.

NullPointerException − keď vstup je null hodnota.

SQLException − keď je chyba počas prístupu na databázu.

`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:

String 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:

int počet objektov v zásobníku A.

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

Vráti počet objektov v zásobníku B.

Návratová hodnota:

int počet objektov v zásobníku B.

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

Odstráni objekt zo zásobníka A na požadovanej pozícii.

Parametre:

i − int pozícia objektu, ktorú chceme odstrániť.

Návratová hodnota:

OWDObject objekt, ktorý sme odstránili.

Hádže:

InterruptedException − keď bol prerušení počas čakania na zápis.

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

Odstráni objekt zo zásobníka B na požadovanej pozícii.

Parametre:

i − int pozícia objektu, ktorú chceme odstrániť.

Návratová hodnota:

OWDObject objekt, ktorý sme odstránili.

Hádže:

InterruptedException − keď bol prerušení počas čakania na zápis.

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

Vráti objekt zo zásobníka A na požadovanej pozícii.

Parametre:

i − int pozícia objektu.

Návratová hodnota:

OWDObject objekt na danej pozícii.

Hádže:

InterruptedException − keď bol prerušení počas čakania na zápis.

`public statit OWDObject readB(int i)`

Vráti objekt zo zásobníka B na požadovanej pozícii.

Parametre:

i − int pozícia objektu.

Návratová hodnota:

OWDObject objekt na danej pozícii.

Hádže:

InterruptedException − keď bol prerušení počas čakania na zápis.

`public void setlogl(String level)`

Nastavuje úroveň logovania.

3.1.2 Trieda OWDFieldSpecifier

Trieda slúži na prácu s elementmi šablóny a záznamu o toku.

Konštruktor

`public OWDFieldSpecifier()`

Vytvára novú inštanciu triedy.

Metódy

`public void setFieldSpecifierOwd(byte[] buf, int position)`

Nastaví element šablóny a jeho pozíciu v zázname o toku. Metóda si sama zistí či sa jedná o enterprise element.

Parametre:

buf − byte [] element, ktorí chceme nastaviť.
position − int pozícia elementu v zázname o toku.

3.1.3 Trieda OWDTemplateCache

Zásobník na ukladanie šablón IPFIX správy pre účely merania jednosmerného oneskorenia.

Konštruktor

`public OWDTemplateCache()`

Vytvára novú inštanciu triedy.

Metódy

`public void addTemplate(IPFIXTemplateRecord templ,InetAddress ip, long sourceID)`

Pridá záznam šablóny do zásobníka podľa exportéra, ktorý je identifikovaný IP adresou a jeho identifikátorom.

Parametre:

templ − IPFIXTemplateRecord záznam šablóny.
ip − InetAddress IP adresa exportéra.
sourceID − long identifikátor exportéra.

`public int getIPFIXTemplateCacheCount()`

Zistí počet šablón v IPFIXTemplateCache.

Návratová hodnota:

int – počet šablón.

`public Hashtable` **`getTemplates(InetAddress ip, long sourceID)`**

Vráti všetky šablóny v zásobníku pre daný exportér.

Parametre:

ip – InetAddress IP adresa exportéra.
sourceID – long identifikátor exportéra.

Návratová hodnota:

Hashtable – zoznam šablón v zásobníku pre daný exportér.

`public boolean` **`contains(int templID, InetAddress ip, long sourceID)`**

Zistí či sa šablóna identifikovaná svojim ID nachádza v zásobníku pre daný exportér.

Parametre:

templID – int identifikátor šablóny.
ip – InetAddress IP adresa exportéra.
sourceID – long identifikátor exportéra.

Návratová hodnota:

boolean – true, ak sa šablóna nachádza v zásobníku, false opačne.

`public IPFIXTemplateRecord` **`getByID(int templID, InetAddress ip, long sourceID)`**

Vráti šablónu v zásobníku podľa jej ID a daný exportér.

Parametre:

templID − int identifikátor šablóny.

ip − InetAddress IP adresa exportéra.

sourceID − long identifikátor exportéra.

Návratová hodnota:

IPFIXTemplateRecord − záznam šablóny, null ak sa taká šablóna v cache nenachádza.

3.1.4 Trieda OWDTemplateRecord

Trieda reprezentujúca záznam šablóny.

Konštruktor

`public IPFIXTemplateRecord()`

Vytvára novú inštanciu triedy.

`public IPFIXTemplateRecord(int fieldCount)`

Vytvára novú inštanciu triedy.

Parametre:

fieldCount − int počet polí šablóny.

`public IPFIXTemplateRecord(int templateID, int fieldCount)`

Vytvára novú inštanciu triedy.

Parametre:

templateID – int identifikátor šablóny.

fieldCount – int počet polí šablóny.

Metódy

`public void` **`addField(short elementID, short fieldLength)`**

Vytvorí a pridá pole do šablóny.

Parametre:

elementID – short identifikátor elementu (pole šablóny).

fieldLength – short dĺžka elementu v bajtoch.

`public void` **`addField(short elementID, short fieldLength, int enterprise)`**

Vytvorí a pridá pole do šablóny.

Parametre:

elementID – short identifikátor elementu (pole šablóny).

fieldLength – short dĺžka elementu v bajtoch.

enterprise – int enterprise číslo poľa.

`public void` **`addField(byte[] data)`**

Vytvorí a pridá pole do šablóny.

Parametre:

data – byte[] pole bajtov poľa šablóny.

`public void` **`addField(FieldSpecifier field)`**

Pridá existujúce pole do šablóny.

Parametre:

field − FieldSpecifier pole šablóny.

`public FieldSpecifier getField(int index)`

Vráti referenciu na pole zo šablóny podľa indexu v zozname.

Parametre:

index − int index poľa.

Návratová hodnota:

FieldSpecifier − pole šablóny.

`public FieldSpecifier getFieldByElementID(int elementID)`

Vráti referenciu na pole zo šablóny podľa identifikátora poľa.

Parametre:

elementID − int identifikátor poľa.

Návratová hodnota:

FieldSpecifier − pole šablóny, null ak také pole v šablóne neexistuje.

`public List<FieldSpecifier> getFields()`

Vráti zoznam všetkých polí v tejto šablóne.

Návratová hodnota:

List<FieldSpecifier> − zoznam polí.

`public int getFieldSpecifierPosition(short elementID)`

Vráti index poľa šablóny podľa jeho identifikátora.

Parametre:

elementID − short identifikátor poľa.

Návratová hodnota:

int − index poľa v zozname polí šablóny, -1 ak sa také pole v šablóne nenachádza.

3.1.5 Trieda OWDObject

Trieda slúži na uloženie hodnôt potrebných pre výpočet jednosmerného oneskorenia v objekte.

Konštruktor

`public OWDObject(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)`

Vytvára novú inštanciu triedy.

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 pektu.

addr – InetAddress adresa od ktorej paket prišiel.

3.1.6 Trieda OWDListener

Trieda robí pred-spracovávanie údajov. Kontroluje a kategorizuje pakety prijaté v zhromažďovacom procese a podľa konfiguračných nastavení ich ukladá v zásobníkoch pre meranie jednosmerného oneskorenia (OWDCache).

Konštruktor

`public OWDListener()`

Vytvára novú inštanciu triedy.

Metódy

`public void setlogl(String level)`

Nastavuje úroveň logovania.

`public void preProcessing(ByteBuffer packet, InetSocketAddress addr)`

Kontroluje správnosť prijatých paketov a porozdeľuje ich medzi zásobníkmi pre meranie owd.

Parametre:

packet − ByteBuffer paket na spracovanie. addr − InetSocketAddress adresa zdroja od ktorého paket prišiel.

Hádže:

UnknownHostException − keď sa nedá zistiť adresa hostu.
InterruptedException − keď je prerušený počas čakania na zápis do zásobníka.
NullPointerException − keď vstup je null hodnota.
SQLException − keď je chyba počas prístupu na databázu.

```
public int testRecordSet(ByteBuffer packet, int index,
InetSocketAddress addr)
```

Kontroluje správnosť prijatých paketov na základe prijatých šablón, záznamov o toku a nastavení v konfiguračnom súbore.

Parametre:

packet − ByteBuffer paket na spracovanie.
index − int index začiatku setu.
addr − InetSocketAddress adresa zdroja od ktorého paket prišiel.

Návratová hodnota:

int − kód pre danú switch-case vetvu. (7 − pre owdStart merací bod, 8 - pre owdEnd merací bod, v inom prípade sa preskočí meranie pre daný paket).

```
public byte[] getTemplateSet(ByteBuffer packet)
```

Metóda získa šablónu z aktuálneho paketu.

Parametre:

packet − ByteBuffer paket na spracovanie.

Návratová hodnota:

byte[] − šablóna.

`public void parseTemplateSet(ByteBuffer template, InetSocketAddress addr)`

Metóda sparsuje celú šablónu a uloží hodnoty potrebné pre výpočet owd.

Parametre:

packet − ByteBuffer paket na spracovanie.
addr − InetSocketAddress adresa zdroja od ktorého paket prišiel.

`public ByteBuffer expandTemplateSet(ByteBuffer template)`

Metóda rozšíri šablónu o polia pre dve owd hodnoty.

Parametre:

template − ByteBuffer šablóna na spracovanie.

Návratová hodnota:

ByteBuffer − rozšírená šablóna.

3.1.7 Trieda OWDFlushCacheABThread

Vlákno, ktoré kontroluje objekty v zásobníkoch pre výpočet jednosmerného oneskorenia. Keď objekt je viac ako 10 sekúnd v zásobníku, dáta sa presunú na pôvodné spracovanie.

Konštruktor

`public OWDFlushCacheABThread()`

Vytvára novú inštanciu triedy.

3.1.8 Trieda Synchronization

Vlákno, ktoré sa správa ako synchronizačný server. Na nastaviteľnom porte čaká na prijatie časových známok na ktoré odpovie svojím lokálnym časom. Pomocou vlákna na meracích bodoch, medzi ktorými sa meria jednosmerné oneskorenie bude zabezpečený synchronizovaný čas, ktorá je nevyhnutná pre správnosť owd hodnôt.

Konštruktor

`public Synchronization(int port)`

Vytvára novú inštanciu triedy.

Parametre:

port − int port na ktorom sa očakávajú časové známky.

Hádže:

SocketException − keď sa vyskytla chyba na nižšej vrstve protokolu.
IOException − Signalizuje vstupno/výstupnú výnimku.

3.1.9 Trieda NetConnect

Vlákno na zachytávanie paketov od meracích bodov. V tejto triede sa určuje typ protokolu (UDP alebo TCP), pomocou ktorého prebieha komunikácia medzi jed-

notlivými časťami meracej architektúry. Zároveň, táto trieda je zodpovedná za poskytovanie paketov pre výpočet jednosmerného oneskorenia.

Konštruktor

`public NetConnect(int port, int bufferSize)`

Vytvára novú inštanciu triedy.

Parametre:

port − int port na ktorom sa očakávajú pakety. bufferSize − int veľkosť zásobníka pre pakety.

Hádže:

IOException − Signalizuje vstupno/výstupnú výnimku.

`public NetConnect(int port)`

Vytvára novú inštanciu triedy so štandardnou veľkosťou zásobníka pre prijaté pakety.

Parametre:

port − int port na ktorom sa očakávajú pakety.

Hádže:

SocketException − keď sa vyskytla chyba na nižšej vrstve protokolu.
IOException − Signalizuje vstupno/výstupnú výnimku.

4 Preklad programu

4.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe bakalárskej práce.

Sú k dispozícii tieto zdrojové texty:

- balíček sk.tuke.cnl.bm:
  - ACPIPFIXTemplate.java
  - Filter.java
  - InetAddr.java
  - InvalidFilterRuleException.java
  - Sampling.java
  - SimpleFilter.java
  - Templates.java
- balíček sk.tuke.cnl.bm.JXColl:
  - Config.java
  - Debug.java
  - IJXConstants.java
  - IpfixElements.java
  - JXColl.java
  - NetConnect.java
  - NetXMLParser.java
  - OutputCache.java
  - PacketCache.java
  - PacketListener.java
  - PacketObject.java
  - RecordDispatcher.java
  - TCPMultiServerThread.java

XMLNode.java

XMLPacket.java

- balíček sk.tuke.cnl.bm.JXColl.IPFIX:

FieldSpecifier.java

IPFIXDataRecord.java

IPFIXMessage.java

IPFIXOptionsTemplateRecord.java

IPFIXSet.java

IPFIXTemplateCache.java

IPFIXTemplateRecord.java

- balíček sk.tuke.cnl.bm.JXColl.OWD:

OWDCache.java

OWDFieldSpecifier.java

OWDFlushCacheABThread.java

OWDListener.java

OWDObject.java

OWDTemplateCache.java

OWDTemplateRecord.java

Synchronization.java

- balíček sk.tuke.cnl.bm.JXColl.accounting:

AccountingManager.java

AccountingRecord.java

AccountingRecordsCache.java

AccountingRecordsExporter.java

- balíček sk.tuke.cnl.bm.JXColl.export:

ACPIPFIXWorker.java

ACPServer.java

DBExport.java

PGClient.java

4.2 Požiadavky na technické prostriedky pri preklade

Preklad programu si vyžaduje nasledovnú hardvérovú konfiguráciu:

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

4.3 Požiadavky na programové prostriedky pri preklade

- ľubovoľný operačný systém s podporou Java Virtual Machine (JVM) (Windows XP/Server 2003/Vista/7, GNU/Linux/Debian/Ubuntu alebo Solaris)
- Java Runtime Environment (JRE) verzie 1.6.0_24 a vyššej
- knižnice dodávané na inštalačnom médiu

4.4 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).

4.5 Vlastný preklad

Preklad programu spočíva v nakopírovaní zdrojových súborov a spustení kompilátora jazyka Java s potrebnými parametrami a parametrom classpath nastaveným na prídavné knižnice. Odporúča sa použiť váš obľúbený java IDE, kde stačí jednoducho

nastaviť verziu JDK na 6.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam.

4.6 Opis známych chýb

V programe JXColl v3.6 bola objavená chyba, ktorá sa prejavuje pri ukončovaní TCP spojenia. Táto chyba je spôsobená nevhodnou implementáciou TCP spojenia v exportovacom a zhromažďovacom procese.

5 Zhodnotenie riešenia

Overenie implementácie prinieslo očakávané výsledky, na základe ktorých bola zistená správnosť funkčnosti navrhnutého modulu. Meranie pomocou modulu však prinieslo aj jednu nevýhodu. Keďže dátové záznamy o tokoch sú držané v zásobníkoch istý čas, ktorý zaleží od parametra nastaveného v exportovacom procese, možnosť pri zapnutom module merania jednosmerného oneskorenia vylučuje monitorovanie sieťovej prevádzky v reálnom čase. Z toho dôvodu by bolo v budúcnosti potrebné metódu výpočtu jednosmerného oneskorenia presunúť do takej vrstvy architektúry BasicMeter, ktorá má najmenší vplyv na monitorovanie sieťových charakteristík. Jednou takou možnosťou by bolo premiestnenie výpočtu jednosmerného oneskorenia do WebAnalyzer-a meracieho nástroja, ktorý by údaje potrebné na výpočet hodnôt získaval z databázy. Takto by sa predišlo časovo zdlhavému predspracovávaniu obdržaných dát kolektorom, v dôsledku čoho by nebolo žiadnym spôsobom ovplyvnené vyhodnotenie záznamov o tokoch siete v reálnom čase.

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 práca, KPI FEI TU, Košice

[3]Pekár, A.: Meranie prevádzkových parametrov site v reálnom čase, 2009, Bakalárska práca, KPI FEI TU, Košice

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