AnalyzerBgp

Progetto Gestione di Reti
anno accademico 2009/10

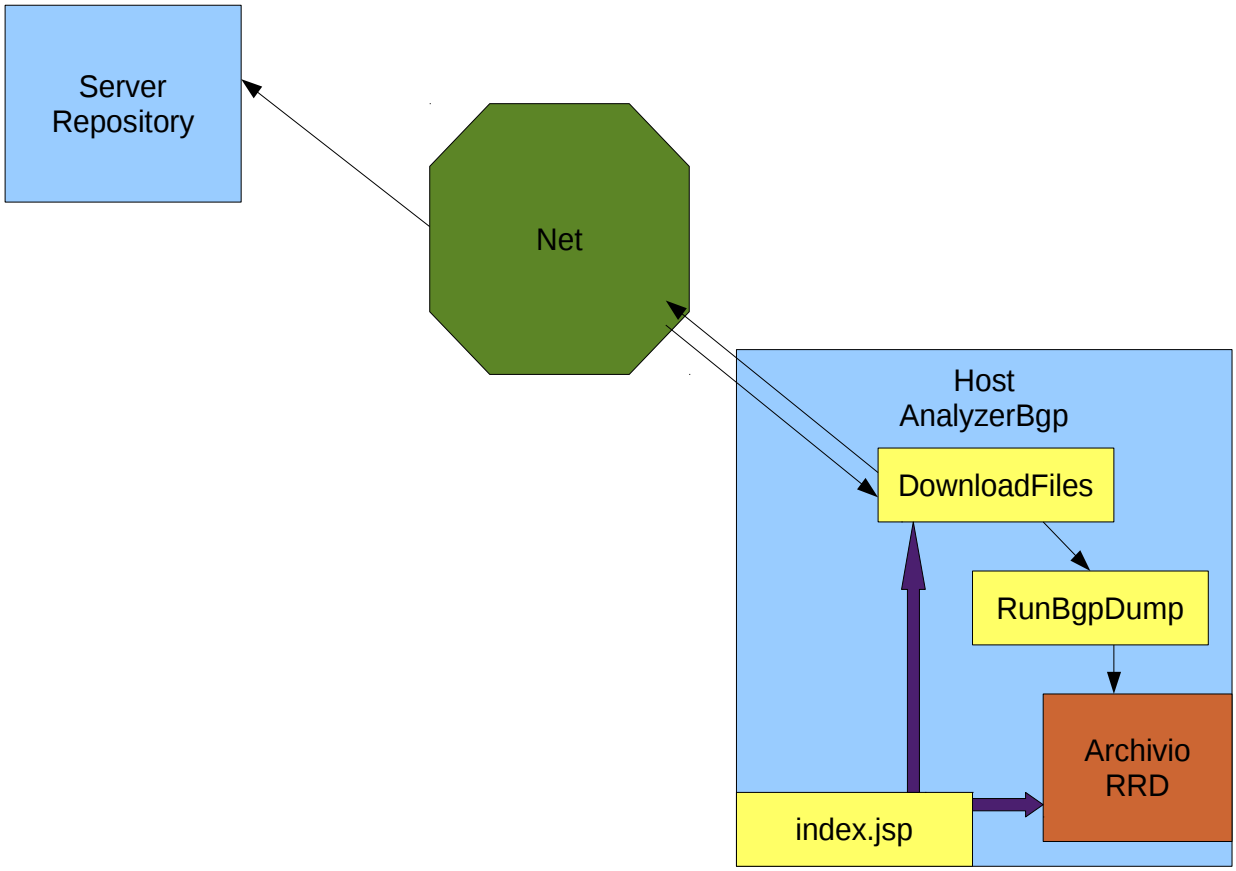
Sulas Domenico

Indice

1. Breve descrizione

AnalyzerBgp è una semplice applicazione che analizza dei pacchetti BGP e rappresenta le informazioni ottenute tramite un interfaccia web 2.0.

L'applicazione consiste per lo più in un thread (DownloadFiles) a cui si passa l'URL di un server su cui vengono salvati i file in formato MRT, tale thread si occupa di reperire i file e passarli ad un secondo thread (RunBgpDump) che ne analizza il contenuto e produce, con l'ausilio di RRDTool e LibBgpDump, informazioni statistiche. In fine una pagina JSP (index.jsp) si occupa di visualizzare tali informazioni.



2. Avviare l'Applicazione

2.1. Prerequisiti

Il progetto è pensato per girare su Tomcat efunziona in ambiente linux.Per un corretto funzionamento son necessari:

- rrdtool (http://oss.oetiker.ch/rrdtool/)
- bgpdump (http://www.ris.ripe.net/source/)
- tomcat (http://tomcatapache.org/)
- java 6 (sun-java6-jdk: http://java.sun.com/javase/downloads/widget/jdk6.jsp)

2.2. Installazione

1. java (jdk):
   - su una distribuzione Debian/Ubuntu, usando un account con i permessi di "root", è sufficente digitare in una shell «apt-get install sun-java6-jdk»
2. rrdtool:
   - su una distribuzione Debian/Ubuntu, usando un account con i permessi di "root",è sufficente digitare in una shell «apt-get install rrdtool»
3. bgpdump:
   - scaricare http://www.ris.ripe.net/source/libbgpdump-1.4.99.11.tar.gz e compilarlo
   - rendere eseguibile bgpdump (digitare nella shell «chmod 777 ./bgpdump»)
   - copiarlo, con i permessi di root, in /usr/bin/ (digitare nella shell «mv ./bgpdump /usr/bin/»)
4. tomcat:
   - scaricare http://apache.fis.uniroma2.it/tomcat/tomcat-6/v6.0.28/bin/apache-tomcat-6.0.28.tar.gz
   - estrarre il file «tar xvzf apache-tomcat-6.0.18.tar.gz /home/"nomeUtente"/»
5. Del progetto va fatto il deploy, cioè:
   - andare su «http://indirizzo-ip-server:8080/manager/html/list» in «WAR file to deploy», nello specifico in «Select WAR file to upload», selzzionare il file «AnalyzerBgp.war» e cliccare su «deploy»
   - usando Eclipse e l'apposito plugin «Dynamic Web Project» (http://download.eclipse.org/webtools/downloads/), e sufficente esportare il progetto come War File nella cartella webapps di tomcat

2.3. Usare AnalyzerBgp

2.3.1.1. Avvio

Dopo aver avviato Tomcat ed aver fatto il deploy del progetto, per avviare l'applicazione e' sufficiente aprire un browser ed andare al indirizzo "http://indirizzo-ip-server:8080/AnalyzerBgp/Analyzer".

2.3.1.2. Analizzare pacchetti di un determinato gruppo

L'applicazione funziona su una qualsiasi raccolta di dati strutturata in modo simile a quella di default. Per default vengono analizzati i pacchetti raccolti dai membri di LINX (http://www.ris.ripe.net/peerlist/rrc01.shtml), questi son reperibili al indirizzo http://data.ris.ripe.net/rrc01.

Più precisamente i nomi di file ricercati devono essere nella forma updates.yyyymmdd.hhmm.gz e devono esser contenuti (o meglio raggruppati) in directory yyyy.mm/ . Queste a loro volta devono esser contenute in una directory "principale" X.

Per iniziare ad analizzare una nuova raccolta di dati è sufficiente inserire nell'apposita form "Insert new Source" l'URL della directory principale.

Esempio:

inserendo nella form l'URL http://a.b.c/X l'applicazione considera X come la directory principale, e quindi alle 11:17 in data 1/1/2100 cerca di ottenere il file updates.21000101.1115.gz contenuto nella cartella 2100.11/ contenuta in X, la cui URL deve essere http://a.b.c/X/2100.11/updates.21000101.1115.gz!

2.3.1.3. Scelta repository

Quando si stanno analizzando i pacchetti provenienti da più repository la form "Select one" permette di scegliere quali risultati visualizzare. Naturalmente visualizzare i risultati ottenuti da repository, non influenza e/o interrompe il processo di analisi sui repository restanti.

2.3.1.4. Fermare l'analisi

Tramite l'apposita form "Stop Monitoring" si ferma il thread, e quindi il relativo processo di analisi, associato al URL selezzionata. Questo è l'unico modo con cui l'utente può terminare l'analisi del repository, in quanto anche se la servlet di gestione viene fermata i threads continuano a lavorare in background. Naturalmente allo spegnimento di Tomcat, o al undeploy del progetto i threads vengono arrestati in automatico.

ATTENZIONE: da quando si invia il segnale di stop il thread può impiegare anche 5 minuti a fermarsi. Ciò è dovuto alla sua struttura, infatti dopo aver scaricato un file il tread si addormenta per 5 minuti per dare il tempo al Server di uppare un nuovo file da analizzare.

3. Struttura del progetto

3.1. Classi del progetto

AutonomSystem.java

è una classe che immagazzina le informazioni sugli AS. In particolare conta i pacchetti BGP inviati da ogni AS e tiene traccia degli AS più attivi (quelli che inviano più pacchetti BGP).

NextHopInfo.java

è una classe per immagazzinare le informazioni sui router di next hop, cioè sui router verso cui instradare il traffico per una determinata destinazione. Per ognuno di questi router, la classe tiene traccia del numero di rotte per cui gli viene instradato il traffico.

OriginInfo.java

è una classe per immagazzinare informazioni sul origine dei pacchetti. In particolare la classe tiene traccia del numero di pacchetti IGP, EGP ed INC. Visto che un pacchetto può essere generato al interno di un Autonom System o provenire da un Autonom System esterno, conta i pacchetti generati da un router interno al AS il numero dei pacchetti provenienti da un altro AS ed il numero dei pacchetti per cui non si può stabilire la provenienza.

Analyzer.java

è una Servlet il cui compito principale è quello di smistare le richieste http (solamente le GET e le POST) verso una pagina JSP adeguata. Inoltre la classe avvia i threads che si occupano di scaricare i file contenenti i pacchetti da analizzare.

Repository.java

la classe gestisce il file ("./webapps/AnalyzerBgp/Data/SitiNoti.txt") in cui vengono salvate le URL dei repository dei file contenenti i pacchetti da analizzare.

ServletListener.java

classe che inizializza, nel contesto della servlet, un vettore contenete i threads attivi. Viene usata per garantire che in sessioni diverse non vengano lanciati threads gia esistenti.

DownloadFiles.java

è un thread che dopo aver creato un archivio RRD, si occupa di contattare il server che ospita i file contenenti i pacchetti da analizzare e scaricarli. Inoltre per ogni file ottenuto lancia un 2° thread ("RunBgpDump.java") che analizza i pacchetti. Fatto ciò il thread si ferma per 5 minuti per attendere che venga uppato il nuovo file da richiedere.

ATTENZIONE:

- i thread, una volta avviato, rimane in esecuzione finché non viene sollevata un eccezione o viene fermato esplicitamente. Ciò avviene quando, tramite l'apposita form, si richiede di fermarne il monitoraggio del repository a cui il thread è associato.
- Il thread richiede file “vecchi di 3 ore” sia perchè i server, su cui si appoggia, a loro volta uppano i file con un ritardo di 2 ore che per evitare “buchi” di informazioni. Infatti visto che i due host non son sincronizzati, può capitare di richiede file che non sono ancora stati uppati e la serie di richieste che si generano, spesso vanno ad influire col corretto aggiornamento del archivio RRD.

RunBgpDump.java

è un thread che tramite "libbgpdump", nello specifico usa l'eseguibile distribuito con la libreria "bgpdump", analizza i file contenenti i pacchetti BGP. Passa le informazioni ottenute alle classi appropriate e aggiorna lo specifico archivio RRD precedentemente creato.

ThreadException.java

è un classe che implementando Thread.UncaughtExceptionHandler, permette ai vari threads di sollevare eccezzioni a tempo di esecuzione.

3.2. Pagine JSP del progetto

Welcome.jsp

è la pagina di benvenuto. Permette di scegliere un repository già noto da di monitorare, scegliere un repository per cui fermarne il monitoraggio e di inserire l'URL di un nuovo repository.

Error.jsp

è la pagina designata alla visualizzazione degli errori. In particolare mostra gli errori di comunicazione con i repository. Permette di scegliere un repository già noto da di monitorare, scegliere un repository per cui fermarne il monitoraggio e di inserire l'URL di un nuovo repository.

Index.jsp

è la pagina principale del progetto. Visualizza le informazioni statistiche generate dai thread del applicazione. Come le altre pagine permette di inserire nuovi repository, di visualizzare le informazioni di un repository già sotto monitoraggio e di terminare il monitoraggio di un repository.

3.3. Script usati nel progetto

Ho scelto di inserire tutte le chiamate ad RRDTOOL in script BASH per via della “notevole” quantità di istruzioni, necessarie sopratutto a generare i grafici, che a mio parere mal si prestano ad esser eseguite tramite una `Runtime.getRuntime().exec();`

CreaRRD.sh

script bash con cui viene generato l'archivio RRD. In particolare crea un archivio in cui viene registrato il numero dei pacchetti analizzati. In particolare per ogni pacchetto analizzato, vengono registrati nel archivio anche il:

- numero di pacchetti del tipo Open
- numero di pacchetti del tipo Keep Alive
- numero di pacchetti del tipo Notify
- numero di pacchetti del tipo Change State
- numero di pacchetti del tipo Table Dump
- numero di pacchetti del tipo Update, per cui vengono registrate anche
    1. numero di richieste di update
    2. numero di richieste di withdraw

L'archivio si aspetta nuovi dati ogni 5 minuti.

AggiornaRRD.sh

script bash con cui viene aggiornato l'archivio RRD.

DisegnaRRD.sh

script bash che ricava dei grafici dall'archivio RRD. Si deve tener presente che la funzione usata («rrdtool graph») non rappresenta dati «parziali» e quindi in fase di avvio, i grafici creati dallo script possono risultare vuoti nonostante i thread appositi abbiano correttamente reperito ed analizzato i file necessari alla loro creazione. In particolare i grafici, riguardanti un nuovo repository, rimarranno vuoti per almeno 5 minuti.