DNS Proxy Filter

Gestione di Reti 2016/2017

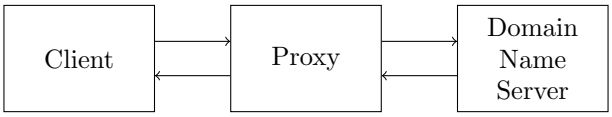
Matteo Bernabito
bernabito.matteo@gmail.com

Luglio 2017

Indice

1 Introduzione

Il progetto riguarda un semplice proxy server DNS sviluppato in Java.
L'applicativo sviluppato, oltre che funzionare come semplice intermediario tra il client e il server DNS finale, permette anche di bloccare una lista di domini specificati in un apposito file, impedendone la risoluzione. Questa funzionalità può permettere, ad esempio, di bloccare tutti i domini contenenti pubblicità a livello di risoluzione, fornendo una lista adeguata.



Dislocazione del proxy all'interno del sistema.

1.1 Il protocollo DNS

Il *Domain Name System* è il sistema utilizzato in ambito di rete per la risoluzione di nomi di host in indirizzi IP. Il sistema e il protocollo da esso utilizzato è ampiamente documentato a partire da RFC 1035. Nel protocollo vengono specificati nei dettagli:

- Formato dell'intestazione del messaggio
- Formato del corpo del messaggio
- Livello di trasporto

Per quanto riguarda il livello di trasporto, dal punto di vista delle query, è previsto il supporto sia per datagram UDP che flussi TCP. Detto ciò, viene fatto notare che nonstante ci sia il supporto al trasporto con protocolli orientati alla connessione come TCP, i datagram UDP sono preferibili in quanto hanno un minore overhead e performance migliori. Per questo motivo, si usa TCP solo quando la grandezza delle richieste supera il limite dei 512 bytes (di sola parte payload) stabilito per i datagram. Indipendentemente dal protocollo, la porta dei server DNS è la numero 53.

1.2 Semplificazioni di progetto

Durante la progettazione e lo sviluppo del progetto ho deciso di fare alcune assunzioni e semplificazioni:

- Il proxy server supporta solo richieste DNS in UDP.
- Il proxy server non parsa le risposte dei server DNS ma le manda così come arrivano al richiedente, in modo trasparente.

La scelta di supportare solo UDP è legata al fatto che ho notato che quasi tutte le query DNS viaggiano su UDP, e pochissime se non nessuna su TCP. Per quanto riguarda invece la scelta di non controllare la risposta da parte del server DNS finale, posso dire che l'ho deciso perchè mi sembrava un lavoro in più da far fare al proxy server che avrebbe aggiunto solo un inutile overhead.

2 Struttura Applicativo

2.1 Funzionamento

DNS Proxy Filter è un proxy server DNS multithreaded con interfaccia a linea di comando. La logica di funzionamento dell'applicazione è abbastanza semplice ed è divisa nelle seguenti fasi:

1. Parsing dei parametri della linea di comando.
2. Se inserito, lettura del file con i domini da bloccare.
3. Inizializzazione del proxy server UDP.
    3.1. Spawn della thread pool per la gestione concorrente di più connessioni.
    3.2. Binding del socket passivo sulla porta designata.
    3.3. Ciclo che legge i datagram e delega la gestione ai thread della pool.

Una volta delegato un datagram ad un thread della pool, quest'ultimo procede come segue:

1. Tentativo di parsing della richiesta DNS, se fallisce esce.
2. Controlla che il dominio che si tenta di risolvere non sia filtrato, in tal caso risponde con NXDOMAIN ed esce.
3. Inoltra la richiesta al server DNS designato *i-esimo*, se non risponde, la richiesta viene mandata al server *(i+1)-esimo*.
4. Se riceve una risposta, la inoltra al client, altrimenti esce.

2.2 Organizzazione dei file

Le directory e i file di progetto sono organizzati come segue:

- La directory `/src` contiene i file sorgenti.
- La directory `/resources` contiene le risorse di progetto.
- La directory `/test` contiene i file di test di unità e integrazione.

Il progetto è gestito con gradle e quindi sono presenti anche varie directory e file per quest'ultimo quali `build.gradle`, `gradlew`, `gradlew.bat`, `/gradle`.
I sorgenti sono organizzati in package nel seguente modo:

- *bernabito.dnsproxyfilter.cli*: contiene tutto ciò che riguarda l'interfaccia a riga di comando e il main.
- *bernabito.dnsproxyfilter.protocol*: contiene tutto ciò che riguarda il formato dei pacchetti DNS.
- *bernabito.dnsproxyfilter.rfc*: contiene classi statiche ed enumeratori contenenti alcune proprietà descritte negli RFC.
- *bernabito.dnsproxyfilter.server*: contiene le classi che modellano il proxy server e le sue funzionalità.
- *bernabito.dnsproxyfilter.util*: contiene alcune classi di utilità per il logging, etc.

2.3 Librerie utilizzate

Il progetto ha come unica dipendenza la libreria JCommander, utilizzata per il parsing dei parametri nella CLI. Per maggiori informazioni su JCommander visitare la pagina ufficiale della libreria: `http://jcommander.org/`.
Per quanto riguarda invece la parte di testing, sono state usate le librerie JUnit e Mockito.

3 Manuale

3.1 Compilazione e testing

Per compilare e testare l'applicazione eseguire:

```
$ ./gradlew build
```

Il risultato della compilazione sarà in `./build/libs/DNSProxyFilter.jar`.
Per pulire tutto ciò che è stato prodotto dal processo di build eseguire:

```
$ ./gradlew clean
```

Se si è su Windows ricordarsi di digitare `gradlew` anzichè `./gradlew`.

3.2 Utilizzo

Per eseguire il proxy server utilizzare:

```
$ java -jar DNSProxyFilter.jar
```

In questa modalità, si lasciano tutte le impostazioni di default, ovvero il server ascolterà sulla porta 53 e si binderà sull'indirizzo wildcard 0.0.0.0, quindi su tutte le interfacce IPv4, inoltre, i server DNS al quale sono forwardate le richieste sono quelli di Google cioè 8.8.8.8 e 8.8.4.4.
Per cambiare la porta e l'indirizzo di binding usare, per esempio:

```
$ java -jar DNSProxyFilter.jar -b 192.168.1.144 -p 5000
```

Tenere presente che se la porta è nel range delle porte privilegiate, sui sistemi UNIX bisognerà ottenere i privilegi dell'utente *root* per far funzionare tutto correttamente. Per specificare invece dei server DNS diversi usare:

```
$ java -jar DNSProxyFilter.jar -f 208.67.222.222
```

Se si vogliono inserire più server di inoltro basta inserirli dopo il flag `-f` separati da virgola. Per inserire la lista di domini che si vogliono filtrare usare:

```
$ java -jar DNSProxyFilter.jar -d domainstofilter.txt
```

Il file `domainstofilter.txt` dovrà contenere i domini da filtrare, uno per riga. Per altre informazioni sul formato di questo file leggere la sezione 3.3. Normalmente, l'applicativo non stampa nulla sullo standard output bensì crea un file di log `dnsproxyfilter.log` nel quale vengono inseriti alcuni messaggi di notifica su eventi potenzialmenti problematici come ad esempio un server DNS che non risponde oppure una query non valida. Per avere invece un output con più informazioni, tra cui tutti le richieste dei client e il tempo di risposta dei server DNS usare:

```
$ java -jar DNSProxyFilter.jar -v
```

Se invece si vuole il tutto sullo standard output anzichè su file usare:

```
$ java -jar DNSProxyFilter.jar -debug
```

Per avere ulteriori informazioni su tutti i flag utilizzabili usare:

```
$ java -jar DNSProxyFilter.jar --help
```

3.3 Formato file di filtraggio

Il file contenente i domini da filtrare deve contenere un dominio per linea ed eventualmente può contenere delle linee vuote, che verranno ignorate. L'applicativo supporta anche il blocco di un dominio e di tutti i sottodomini usando la notazione con wildcard (esempio: `*.facebook.com`).

Se ad esempio avessimo un file con queste righe:

- `*.facebook.com`
- `*.adsense.com`
- `google.com`

avremmo bloccato `facebook.com` e tutti i suoi sottodomini, come ad esempio `www.facebook.com`. Lo stesso si può dire per il dominio Adsense. Nel terzo caso, invece, dato che non è presente una wildcard, verrebbe bloccato solo il dominio `google.com` ma, ad esempio, sarebbe possibile accedere a `drive.google.com`.

4 Conclusioni

4.1 Limiti

L'applicazione funziona bene in base a quanto si era prefissato come obiettivo tuttavia presenta alcuni limiti:

- Non supporta DNS su TCP come descritto nella sezione 1.2.
- Non supporta DNSSec.
- L'applicazione richiede un riavvio se si vuole cambiare la lista dei domini filtrati.

4.2 Parti migliorabili

Aldilà dei limiti descritti, ci sono delle parti del codice che si potrebbero migliorare come ad esempio quelle nel package *bernabito.dnsproxyfilter.server*.
Facendo un pò di refactoring si può disaccoppiare il comportamento della classe anonima *ProxyRequestWorker* dal tipo *DatagramPacket* in modo da rendere possibile e facile la scrittura di un eventuale proxy server TCP. Inoltre, a parere mio sarebbe molto interessante aggiungere anche una opzione per fare *caching*, per un certo intervallo di tempo, delle risposte DNS. In questo modo si potrebbero ottenere risposte alla query DNS molto più rapide in certe casistiche dove le query riguardano sempre gli stessi domini.