Network Monitor
Gestione di Rete 2016/2017

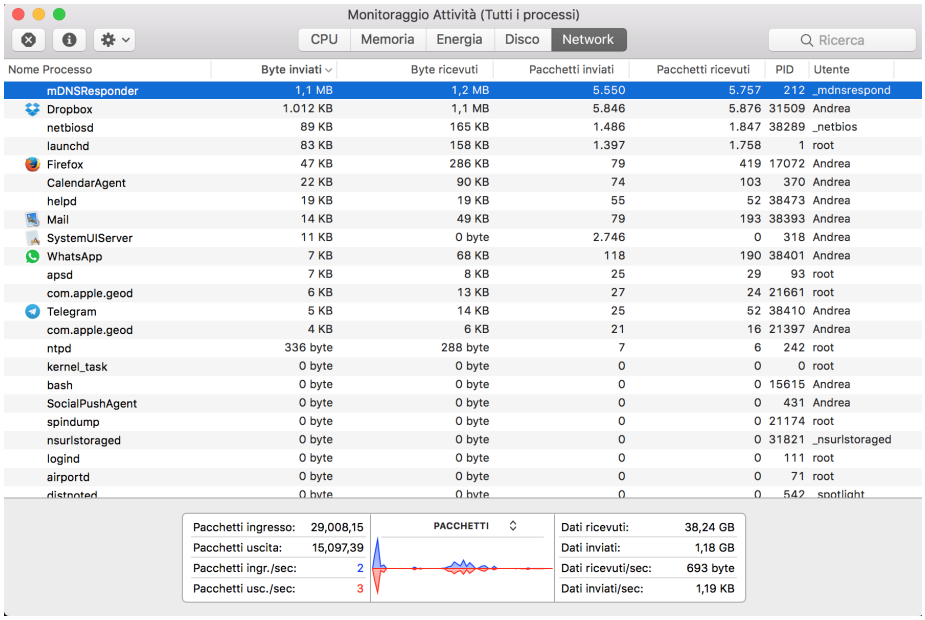
Marco Bianucci
Nicolas Emilio Lupone
Andrea Poggi
Stefania Tola

29 giugno 2017

Indice

1 Introduzione

Lo scopo del progetto è quello di realizzare un'applicazione che permetta di monitorare la quantità di pacchetti e di bytes scambiati sulla rete dai processi in esecuzione all'interno di una macchina.
L'ispirazione del progetto è stata fornita dall'utility *Monitoraggio Attività* presente su MacOSX.



Il progetto è stato realizzato unicamente per sistemi Linux, interamente in linguaggio **C** e con un'interfaccia a riga di comando.

1.1 Ipotesi e semplificazioni

Al fine di non complicare eccessivamente l'implementazione, sono state prese le seguenti decisioni:

- Si conteggiano soltanto i traffici UDP e TCP, in quanto sono quelli dominanti nel normale traffico.
- Si classificano come "irriconoscibili" tutte le altre tipologie di pacchetti, riportandone un opportuno conteggio.
- Il conteggio dei pacchetti totali è relativo alla somma del traffico di tutte le applicazioni rilevate dall'avvio del programma non solo di quelle visualizzate nel riepilogo.

2 Architettura applicazione

2.1 Librerie utilizzate

2.1.1 Librerie di terze parti

- libpcap
  Utilizzata per intercettare i pacchetti dalla rete.
- libprocps
  Utilizzata per accedere alle informazioni dei processi in esecuzione sulla macchina.

2.1.2 Librerie interne

- intQueue.h
  Libreria per gestire un array circolare di interi.
- arrQueue.h
  Libreria per gestire un array circolare di informazioni relative ai pacchetti.
- int_list.h
  Libreria per la gestione di una lista di interi.
- mymacro.h
  Insieme di macro di utilità generiche.

2.2 Strutture dati

Di seguito una descrizione delle principali strutture dati utilizzate:

- process_t *process_list
  Array utilizzato per mantenere le informazioni e conteggi relativi a ciascun processo. L'array viene allocato dopo aver letto il **pid_max**, in modo da poter utilizzare il PID di ciascun processo come indice.
- queueA_t queuePkt
  Array circolare che mantiene le informazioni relative ai pacchetti analizzati ma non ancora collocati nella struttura di cui sopra, in quanto non si dispone ancora del PID del processo interessato.
- queueA_t queueMiss
  Array circolare avente le stesse caratteristiche di quello sopra citato, utilizzato per mantenere i pacchetti a cui non è stato possibile associare immediatamente un PID. Questi saranno ricontrollati in un secondo momento.
- queueI_t queueDead
  Array circolare in cui vengono posti i PID dei processi risultati non più attivi nell'ultima scansione, per cui è quindi necessario riazzerare il conteggio di pacchetti e bytes.

2.3 Struttura del codice

L'applicazione è composta da 4 thread cooperanti, ognuno dei quali si occupa di svolgere un compito ben preciso.

- **Main**
  Si occupa di catturare i pacchetti, analizzarli e di inserirli in una coda.
- **Thread *Put***
  Preleva dalla coda i pacchetti inseriti dal main e li smista in base al processo coinvolto.
- **Thread *Scan***
  Si occupa di identificare quali porte sono utilizzate dai processi in esecuzione.
- **Thread *Demon***
  Ottiene le informazioni sui processi in esecuzione sulla macchina e stampa a video la lista dei processi con il relativo traffico.

2.4 Funzionamento

Al lancio dell'applicazione viene avviato il thread *main*, che si occupa di inizializzare le varie strutture dati necessarie al corretto funzionamento, mettere in esecuzione i vari thread e catturare i pacchetti.
All'arrivo di un pacchetto, questo viene catturato con l'ausilio della libreria **pcap** e analizzato per ottenere le informazioni di cui abbiamo bisogno. L'analisi del pacchetto produce le seguenti informazioni:

- porta sorgente
- porta destinazione
- protocollo di trasporto
- direzione
- dimensione payload

Queste informazioni vengono quindi inserite in un'apposita coda, dalla quale verranno successivamente prelevate dal thread *Put*.
Mentre viene creata la coda, il thread *Scan* si occupa di generare una tabella di associazioni fra le porte aperte e i processi proprietari, utilizzando il comando **netstat**.
Inoltre, grazie alla libreria **libprocps**, il thread *Demon* aggiorna la lista relativa ai processi in esecuzione, eliminando quelli morti.
Man mano che la coda delle informazioni relative ai pacchetti viene riempita, il thread *Put* si occupa di prelevarle e, grazie all'infrastruttura creata dal thread *Scan*, trova il PID del processo responsabile e aggiorna i relativi conteggi di byte e pacchetti inviati o ricevuti.

Nel caso in cui il thread *Put* non riesca a identificare il PID a cui appartiene il pacchetto, le informazioni a lui relative vengono poste in una seconda coda, che sarà nuovamente controllata dallo stesso thread dopo l'aggiornamento della tabella di corrispondenza sopra citata da parte del thread *Scan*.
Se infine questi pacchetti non sono riconducibili a nessun processo, vengono conteggiati come pacchetti sconosciuti.
Quando necessario, il thread *Put* si occupa inoltre di riazzerare il conteggio di pacchetti e bytes relativo ai processi non più in esecuzione.
Al fine di velocizzare il più possibile l'esecuzione e limitare così la possibile perdita di pacchetti, si è cercato di ottimizzare l'applicazione in modo che questa non necessitasse mai l'utilizzo della mutua esclusione.

3 Manuale Utente

Insieme al sorgente dell'applicazione è stato fornito un makefile per facilitarne la compilazione.
Per utilizzare l'applicazione è quindi necessario eseguire i seguenti comandi:

- **make compile**
  Si occupa di compilare i sorgenti, creando l'eseguibile finale.
- **make clean**
  Si occupa di cancellare gli eventuali file oggetto .o e l'eseguibile di precedenti compilazioni, per obbligare la rigenerazione globale dell'applicazione.
- **./network_monitor**
  Applicazione vera e propria.

Per lanciare l'applicazione sono necessari i permessi di **root**, in quanto richiesti dalla libreria pcap, ed è necessario specificare il nome dell'interfaccia da cui si desidera catturare i pacchetti.
Le interfacce presenti nel PC in uso sono visualizzabili con in comando **ifconfig**.

Quindi, per lanciare l'applicazione, è necessario eseguire il seguente comando:

sudo ./network_monitor ⟨**interfaccia**⟩

```
Pacchetti non Riconosciuti: 0   Pacchetti Totali: 15370
-------------------------------------------------------------------------------------------------------------------------
PID             KbyteS          KbyteR          pktS            pktR            USER            NAME
-------------------------------------------------------------------------------------------------------------------------
-               0               0               3               3               -               data-pkt unassigned!
1               0               0               1               3               root            systemd
2               0               0               1               1               root            kthreadd
1580            0               0               1               1               systemd         systemd-resolve
6650            0               0               37              41              nicolas         thunderbird
6793            1               0               6               6               nicolas         gitkraken
6859            0               0               11              11              nicolas         gitkraken
10813           131             10738           6608            8530            nicolas         firefox
12231           1               4               10              7               nicolas         code
12293           5               12              31              22              nicolas         code
17182           0               0               3               2               nicolas         ssh
```

4 Conclusioni

4.1 Test effettuati

L'applicazione è stata testata con successo sui seguenti sistemi operativi:

- Debian 8.8 (Jessie), 64 bit
- Mint 18.1 (Serena), 64 bit
- Ubuntu 16.04, 64 bit
- Ubuntu 17.04, 64 bit

4.2 Limitazioni conosciute

L'applicazione, al suo stato attuale, presenta le seguenti limitazioni:

- A livello rete viene riconosciuto solamente il protocollo IPv4.
- A livello trasporto vengono riconosciuti solamente i protocolli TCP e UDP
- Viene preso in considerazione il solo traffico unicast.
- Il traffico relativo a un processo che nasce e muore in un tempo inferiore a quello impiegato per l'aggiornamento delle strutture dati e la visualizzazione potrebbe non essere identificato.

5 Riferimenti e recapiti

Bianucci Marco	464134	bianucci.marco@virgilio.it
Lupone Nicolas Emilio	490591	luponenicolas@yahoo.it
Poggi Andrea	524864	andrea.poggi5@gmail.com
Tola Stefania	293145	stefania.tola@gmail.com