Implementazione sniffer di rete con libpcap

progetto Gestione di Rete 2013

Quartarone Giuseppe

Il presente documento mira a descrivere l'implementazione di uno sniffer di rete utilizzando la libreria libpcap. Il progetto prevede la cattura del traffico sia da interfaccia di rete che da file, fornendo informazioni utili per anali successive. In oltre è stato previsto il supporto ad rrdtool il quale prevede la creazione di grafici riportanti i risultati delle anali effettuate.

Il progetto è stato sviluppato con il linguaggio C, su un sistema linux con kernel 2.6.43.8-1.fc15.i686.PAE. Per la compilazione è necessario installare oltre alla libreria libpcap anche le librerie per il supporto ad rrdtool. La cattura da interfaccia di rete è possibile solo se si dispongono dei diritti di amministratore, dato che la libreria ha la necessità di interfacciarsi con la scheda di rete. In oltre, per l'esecuzione, è previsto il passaggio di opportuni parametri di configurazione che possono essere visionati lanciano lo sniffer con l'opzione “-h”. Altre configurazioni piu` dettagliate possono invece essere impostate nel codice attraverso le opportune macro. Un ultimo appunto va fatto per il supporto ad rrdtool, il quale necessita dei permessi di scrittura, per l'utente “nobody”, per la generazione del database RRD. Oltre al codice dello sniffer è presente uno script in bash per la creazione dei grafici per visionare le analisi effettuate.

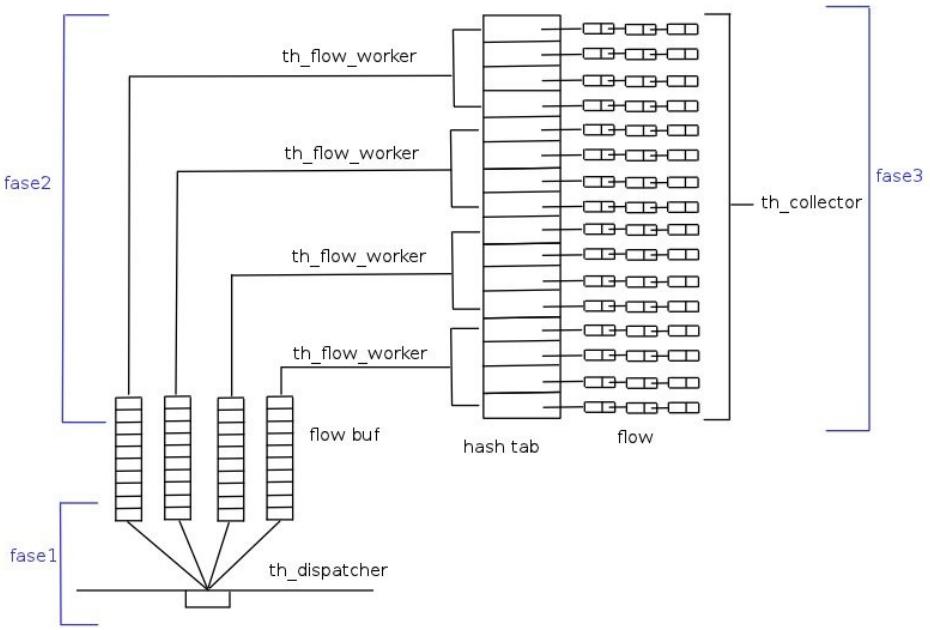
Di seguito verrà presentata la struttura del progetto che può essere suddivisa in tre fasi fondamentali. L'esposione non si addentrerà nella descrizione dettagliata delle funzioni che compongono il progetto, piuttosto verranno presentate le strutture dati ed il loro utilizzo.

Come mostrato nell'immagine sottostante, le fasi fondamentali sono:

1. cattura del traffico;
2. scomposizone dei pacchetti
3. ricostruzione del flusso

Per la fase di cattura è necessario configurare la libreria pcap per istruirla all'ascolto sull'interfaccia di rete o per la lettura da file, i pacchetti catturati vengono inseriti all'interndo delle code. La scelta della coda, per garantire una maggiore distriburzione del carico di lavoro, è effettuata attraverso l'operazione modulo sul tempo contentuo nel pacchetto.

La fase di scomposizione o spacchettamento dei pacchetti è affidata ai thread worker (*th_flow_worker*) che si occupano di prelevarli dalla coda, scomporli per riempire un'opportuna struttura dati con le informazioni ritenute piu` importanti, ed inserire il risutato in una tabella.



Ovviamente tra le informazioni devono essere presenti almeno quelle necessarie ad identificare un flusso. Nell'implementazione presentata un flusso è identificato dalla quintupla:

<ip sorgente, ip destinazione, porta sorgente, porta destinazione, protocollo>

L'hash della quintupla viene utilizzato per inserire le informazioni all'interno della tabella dei flussi, che altro non è che una tabella hash.

Utilizzando un opportuno accorgimento, come visto a lezione, si può aumentare l'efficienza della tabella hash garantendo ad ogni thread che inserisce dati, un opportuna finestra di lavoro. Questo è possibile impostando la dimensione della tabella tale che sia un multipolo del numero dei thread che gestiscono le code. In questo modo si riduce drasticamente il numero di conflitti per gli accessi concorrenti alla tabella.

La terza fase, quella di ricostruzione del flusso invece è affidata ad un altro thread, *th_collector*, il quale scorre periodicamente tutta la tabella alla ricerca di flussi da eliminare ripulendo cosi` l'intera tabella. In realtà nel caso in esame la fase di ricostruzione non è implementata a pieno, infatti la tabella contine frammenti di flusso che andrebbero riassemblati tutti per riscostruire l'intero flusso. Il *th_collector* invece di ricostruire l'intero flusso, elimina i frammenti inserendoli in una sorta di “cimitero”, da cui poi si potrebbe procedere al riassemblaggio vero e proprio. Questa fase non è stata implementata oltre che per motivi di semplicità, anche perchè non vengono effettuate analisi sul flusso.

Dall'analisi effettuata si evince che la tabella hash risulta essere una risorsa ad accesso concorrente e nel caso in esame, viene gestita con l'uso delle pthread_spin_lock e pthread_spin_unlock. Queste risultano avere una maggiore efficienza, rispetto ai semafori mutex, nei casi in cui i tempi di attesa previsti sono di durata molto breve. Caso perfettamente ralistico per l'implementazione di uno sniffer per la cattura del traffico di rete.

Durante la fase di scomposizione viene eseguita in parallelo la raccolta di alcune informazioni da passare al supporto rrdtool per la generazione dei grafici tramite l'utilizzo dello script bash.

Molte delle scelte implementative sono state dettate dalla semplicità, una fra tutte la mancanza della gestione dei segnali per la terminazione corretta di tutte le entità in gioco. Questo perché lo scopo del progetto è quello di mettere in evidenza oltre che l'uso della libreria libpcap, quello della gestione delle informzioni raccolte e le comunicazioni presenti fra le vari entità che compongono l'intero progetto. Durante la fase di progettazione ho tenato di strutturare il codice in modo che possa essere facilemente estendibile per ulteriori ed eventuali miglioramenti.

Di seguito verrà illustrato un esempio dei grafici generati dopo l'esecuzione dello sniffer fin qui presentato.