Out-of-band signaling

Progetto di Laboratorio di Sistemi Operativi
a.a. 2018-19

Federico Pennino

25 ottobre 2019

1 Introduzione

L'obiettivo del progetto è quello di realizzare un un sistema client-server, in cui i client possiedono un codice segreto (che chiameremo secret) e vogliono comunicarlo a un server centrale, senza però trasmetterlo.
Lo scopo è rendere difficile intercettare il secret a chi sta catturando i dati in transito.
Per prima cosa verrà avviato un supervisor che lancerà k server .
Verranno poi lanciati n client che avranno il compito di connettersi in ordine casuale ai server, inviare il loro **id** e aspettare **secret** ms prima di inviare un nuovo messaggio .
I server dovranno valutare il tempo trascorso tra l'*id* uguali e una volta chiusa la connessione al server stimare il *secret* migliore e inviarlo al supervisor .
Compito del supervisor è quello di stimare il valore migliore per ogni client tra tutti quelli ricevuti dai client .

2 Struttura del progetto

La struttura generale del progetto è la seguente :

```
/
├── include - Contiene i moduli che compongono il progetto
│   ├── client - Modulo del client
│   ├── hash_table - Modulo della hash table dei messaggi (supervisor)
│   │
│   ├── server - Modulo del server
│   ├── supervisor - Modulo del supervisor
│   └── macro.h - Contiene le macro e le consts del progetto
├── src - Contiene i "main" del progetto
│   ├── supervisor.c
│   ├── client.c
│   └── server.c
├── misura.sh - Calcola le statistiche del progetto
├── test.sh - Esegue una batteria di test
└── MakeFile
```

3 Moduli

Ogni modulo è formato da tre file :

- support.h - L'header file del modulo
- support.c - Contiene l'implementazione vera e propria
- Makefile - Costruisce il file oggetto del modulo

3.1 Client

Contiene le funzioni e le strutture usate dal client .
Qui si trovano, ad esempio : le funzioni che generano secret e l'id, inoltre, vi si trovano le funzioni che trasformano l'id in *Network Byte Order* .

3.2 Hash_table

Contiene ciò che serve per manipolare l'hash table in cui vengono salvate le migliori stime fatte dal server .

3.3 Server

Contiene le funzioni e le strutture usate dal server .
Le funzioni maggiormente degne di nota sono :

- *new_client(uint64_t,client*,int,int,uint64_t)* , usata per aggiungere una nuova connessione client-server .
- *handle_client_message(int,int,uint64_t,uint64_t,client*,int,fd_set*)*, usata per gestire i messaggi tra client e server in tutte le altre occasioni.

3.4 Supervisor

Contiene alcune dellee funzioni e le struture utilizzate dal supervisor .

4 src folder

4.1 Client.c

Qui viene definito il comportamento di un client .
Inizialmente si genera un id e un secret e, successivamente, vengono selezionati i server a cui connettersi, l'ordine e il numero di volte in cui farlo .
Il messaggio (l'id) è un **uint64_t** e prima di essere trasmesso viene trasformato in **nbo** .

4.2 Server.c

Ciò che fa il server è aspettare che, sulla socket, i client si connettano e inviino un messaggio; una volta che ciò avviene deve aggiungere al set dei file descriptor quello della nuova connessione .
Quando un client invia un messaggio di terminazione deve, invece, scrivere la stima finale del secred del client sul suo *stderr* . Tutte le connessioni vengono gestite tramite una *pselect* scelta per poter aver un handling non brutale dei segnali .
Per terminare un server è necessario inviare un messaggio di SIGTERM che verrà intercettato da una *sigaction* in modo da avere la chiusura del programma e liberare la memoria in heap .

4.3 Supervisor.c

Il compito del supervisor è quello di avviare k processi, tanti quanti sono i server, e tramite k pipe, aspettare una stima .
Viene, anche qui, scelta la *pselect* per osservare i messaggi in arrivo dal server e i segnali .
Per poter salvare le stime è stata usata come struttura di supporto una hash table in maniera tale da poter ottenere in mutua esclusione l'aggiunta di un nuovo id e la stampa di tutta la hash table .
Una nuova aggiunta alla hash table crea un nuovo processo, esattamente come una nuova richiesta di impressione delle stime .
Alla sua terminazione il supervisor ha il compito di aspettare la terminazione dei thread e dei processi.
Inoltre, viene liberata tutta la memoria allocata .

5 Testing

Il progetto è stato testato sui seguenti sistemi :

- Ubuntu 17.04
- WSL (Windows Subsystem for Linux)
- macOS 10.14 "Mojave"
- Xubuntu 14.10

Il progetto è stato inoltre mandato in test su **valgrind** non dando errori o memory leak .