
Università degli Studi di Pisa

Laboratorio di Sistemi Operativi

A.A. 2018/2019

# Relazione Progetto Laboratorio

*Autore:*
Francesco Kotopulos De Angelis

2 novembre 2019

# Indice

# 1 Panoramica del progetto

Lo scopo del progetto è la realizzazione di un sistema di *out-of-band signaling*, una tecnica di comunicazione nella quale due entità si scambiano informazioni attraverso l'uso di una segnalazione collaterale: nel caso in questione il Server deve stimare un numero "segreto" generato dai Client calcolando il ritardo tra l'invio di un primo messaggio ed il successivo da parte di uno stesso Client.
Per la realizzazione di questo progetto sono state definite tre entità principali:

- Supervisor;
- Server;
- Client.

# 2 Scelte Implementative

## 2.1 Supervisor

Il Supervisor è un'entità unica all'interno del progetto ed è la prima entità ad andare in esecuzione. Al suo avvio si occupa di generare "K" Server come processi distinti con i quali manterrà una connessione tramite pipe anonime per poter ricevere le stime calcolate dai Server stessi. Terminata la fase di inizializzazione il Supervisor rimarrà in ascolto sulla pipe in lettura per la ricezione delle stime da parte dei Server.

## 2.2 Server

Il Server è un' entità avviata dal Supervisor che si occupa di rimanere in attesa di nuove connessioni da parte dei Clients e di stimare i *SECRET*. Ad ogni nuova connessione, il Server crea un Thread che rimane in ascolto sulla socket e attende l'arrivo degli *ID* da parte dei Client. Per ogni lettura sulla socket, il Server stima il *SECRET* facendo la differenza tra il tempo corrente espresso in millisecondi e il tempo dell'ultima lettura effettuata. Con questo sistema di misurazione la stima migliore risulta essere quella più bassa perché è quella che presenta il più basso ritardo di trasmissione. Dopo aver effettuato tutte le letture dalla socket, il Thread del Server termina inviando la stima migliore al Supervisor tramite pipe anonime. Dopo la terminazione del Thread il Server continua ad essere in attesa di nuove connessioni da parte dei Client. La terminazione del processo Server può avvenire soltanto con la ricezione di un segnale *SIGQUIT* da parte del Supervisor. In questo caso il Server chiude la Socket sulla quale è in ascolto e termina regolarmente.

## 2.3 Client

Il Client è un'entità che comunica esclusivamente con i Server.
Al suo avvio riceve tre parametri in ingresso:

- P: numero dei Server che il Client deve scegliere per effettuare la connessione;
- K: numero totale dei Server avviati dal Supervisor;
- W: numero totale dei messaggi che il Client deve inviare ai Server selezionati.

I parametri immessi devono rispettare alcune proprietà: $1 <= p <= k$ , $w > 3p$.
Nel caso in cui i parametri non vengano inseriti correttamente, il Client stampa un errore sullo *stdout*. Dopo aver ricevuto i parametri in ingresso il Client genera un *ID* casuale ed univoco di 32bit ed un numero intero *SECRET* che i Server dovranno stimare. Terminata la fase di inizializzazione il Client seleziona casualmente $p$ Server distinti tra i $k$ disponibili avviati dal Supervisor. Successivamente il Client si connette ai Servers precedentemente selezionati ed invia il proprio *ID* ogni *SECRET* millisecondi ad un Server scelto casualmente fra quelli connessi.

## 2.4 Gestione della memoria

Per il corretto funzionamento del sistema sono state inserite all'interno del progetto alcune strutture dati allocate dinamicamente. Ad ogni possibile terminazione dei processi tutte le strutture dati allocate dinamicamente vengono liberate con opportune *free()* per garantire l'assenza di eventuali memory leak.

## 2.5 Gestione dei segnali

All'interno del Supervisor è stata implementata una gestione esplicita dei segnali. In particolare sono state definite tre variabili *volatile sig_atomic_t:*

- *sigint*: per memorizzare la ricezione di un primo *SIGINT*;
- *sigalrm*: per impostare un timer di 1 secondo dopo la ricezione del primo *SIGINT*;
- *sigquit*: per memorizzare la ricezione di due *SIGINT* consecutivi e procedere alla terminazione del sistema.

Tutte le variabili sono inizializzate a "0". Nel momento in cui il Supervisor riceve un primo *SIGINT*, imposta la variabile *sigint* a "1" e dopo un secondo dalla ricezione del segnale invia un *SIGALRM*. Se in questo lasso di tempo è stato intercettato un secondo *SIGINT*, la variabile *sigquit* viene impostata ad "1" e si procede alla terminazione del

sistema. Nel caso in cui il sistema riceva un *SIGALRM* prima di un secondo *SIGINT*, le variabili *sigint* e *sigalrm* vengono impostate nuovamente a "0" e si riprende la normale esecuzione del programma.
I segnali *SIGINT e SIGALRM* vengono ignorati lato Server per garantire una corretta gestione del segnale da parte del Supervisor. I Servers prevedono una gestione esplicita del segnale *SIGQUIT*, inviato dal Supervisor, che determina la terminazione corretta di tutti i processi Server.

# 3 Compilazione ed Esecuzione

All'interno del progetto è stato inserito un Makefile per la compilazione automatica dei targets predefiniti. Di seguito sono elencati i comandi predefiniti utilizzabili per la compilazione e per l'esecuzione di un ciclo completo di test:

- *make:* compila il progetto generando i file eseguibili;
- *make test:* avvia un ciclo completo di test con un Supervisor, 8 Servers e 20 Clients;
- *make clean:* elimina tutti i file con estensioni *.o *∼ *.a *.log OOB-server-*;
- *make cleanall:* esegue una *make clean* ed elimina anche i file esguibili *supervisor, client, server.*

Nel caso in cui sia necessario effettuare un ciclo di test rapidamente è sufficiente inviare il comando *make test* il quale si occuperà automaticamente di compilare nuovamente i file sorgente non ancora compilati o modificati recentemente.

# 4 Casi di Test e Statistiche

All'interno del progetto sono stati inseriti due script bash per poter testare il corretto funzionamento del sistema e per avere informazioni statistiche sull'errore medio di misurazione.

## 4.1 Script Test

Questo script permette di eseguire un ciclo completo di test, lanciando il Supervisor con 8 Server. Avviati tutti i processi Server, dopo due secondi vengono lanciati 20 Clients a coppie con un'attesa di 1 secondo tra una coppia e l'altra. Ciascun Client viene lanciato con gli stessi parametri *p,k,w*, con i rispettivi valori *5,8,20*. Dopo aver terminato la fase di avvio dei Clients, lo script attende 60 secondi inviando un *SIGINT* al Supervisor ogni 10 secondi. Al termine di questo lasso di tempo lo script invia due *SIGINT* ravvicinati

e il Supervisor entra nella fase di terminazione. Dopo che il Supervisor ha atteso la terminazione di tutti i processi Server, viene lanciato un secondo script, *Misura.sh*, che si occupa di calcolare il numero di stime corrette e l'errore medio sulle stime.

## 4.2 Script Misura

Questo script permette di analizzare i file di log generati dal sistema per calcolare il numero di stime corrette e l'errore medio sulle stime. Una stima $s$ di un Server si definisce corretta se il valore di $s$ si discosta dal *SECRET* realmente generato dal Client, per al più un errore di 25 unità. Per facilitare l'analisi del file di log generato dal sistema, lo script *Misura.sh* cerca nel file di log le stime migliori calcolate dal Supervisor e ne salva il *SECRET STIMATO* e *ID* del Client associato a quella stima in un file *supervisor_last.log*. Al fine di calcolare il numero di stime corrette e l'errore medio sulle stime è necessario analizzare anche il file di log generato dal Client. A questo scopo vengono salvati in un file *client_last.log*, *l'ID* e i rispettivi *SECRET* generati dai Clients.