Progetto Laboratorio di Sistemi Operativi

out-of-band signaling

Andrea Bonanno

Dipartimento di Informatica
Università degli Studi di Pisa
16 Ottobre 2019

Contents

1 Scelte Progettuali

Di seguito vengono presentate le varie scelte progettuali effettuate durante la realizzazione del progetto.

1.1 Struttura del codice

Il codice sorgente è suddiviso nei seguenti moduli:

- `client.c`

  Codice relativo all'esecuzione dei client.

- `server.c`

  Codice relativo all'esecuzione dei server.

- `supervisor.c`

  Codice relativo all'esecuzione del supervisor.

- `hash.c`

  Implementazione di una hash table per la memorizzazione temporanea dei dati da parte del supervisor.

- `utils.c`

  Varie funzioni di utility utilizzate da client, server e supervisor.

1.2 Flusso di esecuzione e threads

I client e il supervisor hanno un flusso di esecuzione costituito da un singolo thread. Ogni server ha un flusso di esecuzione costituito da un main thread, e un thread aggiuntivo viene creato in risposta ad ogni richiesta di connessione da parte di un client che sia accettata con successo; questi thread vengono terminati quando il relativo client ha terminato l'invio dei propri messaggi al server.

1.3 Comunicazione tra processi

La comunicazione tra client e server avviene come da specifiche con l'utilizzo di socket nel dominio AF_UNIX. La comunicazione tra server e supervisor è stata realizzata tramite l'utilizzo di unnamed pipes.

1.4 Gestione dei segnali

I client gestiscono in maniera minimale il segnale SIGPIPE nel proprio main thread. I server gestiscono SIGTERM nel proprio main thread, bloccando ogni altro segnale. Il supervisor gestisce SIGINT nel proprio main thread, bloccando ogni altro segnale.

1.5 Script di misura

Lo script di misura sfrutta un programma *awk* appositamente scritto per svolgere il proprio compito. Questa scelta è stata effettuata con l'intento di massimizzare semplicità e chiarezza dello script.

2 Dettagli sul funzionamento

2.1 Stima del secret

La stima inviata da un server al supervisor per un dato client è calcolata come il minimo intervallo di tempo (in ms) tra due messaggi ricevuti dal server da parte di quel client. La stima finale da parte del supervisor per un dato client è a sua volta calcolata come il minimo tra tutte le stime parziali che i server hanno fornito per quel client. Se un dato client ha almeno una volta selezionato lo stesso server consecutivamente per l'invio del proprio messaggio, il supervisor riuscirà a stimare correttamente il secret di quel client.

2.2 Comunicazione tra supervisor e server

I file descriptors delle pipe unnamed utilizzate per la comunicazione tra supervisor e server sono duplicati a seguito del forking del supervisor, resi *half-duplex* tramite chiusura delle opportune estremità e poi resi disponbili al flusso di esecuzione dei server con la chiamata exec effettuata dal processo appena creato. Il formato dei messaggi che i server inviano al supervisor consiste nel proprio id, l'id del client a cui la stima è associata e il secret stimato. Il supervisor legge i dati dalle pipe utilizzando in loop una chiamata *pselect*.

2.3 Strutture dati del supervisor

Il supervisor accumula i dati ricevuti dai server in una hash table, mantenendo le associazioni tra id del client, numero di server che

hanno fornito la stima e valore del secret. L'hash table viene opportunamente aggiornata al presentarsi di nuovi client id e server id.

2.4 Terminazione dei server

I server restano in attesa di connessioni da parte dei client finchè il supervisor non invia loro un segnale SIGTERM. La ricezione del segnale causa ai supervisor di interrompere l'attesa di connessioni sulla socket, l'invio di un *cancel* a tutti i propri thread attivi e l'attesa della terminazione degli stessi tramite *join*. I threads che ricevono il *cancel* terminano dopo aver inviato la loro migliore stima corrente ed aver eseguito azioni di cleanup.

2.5 Handlers dei segnali

Le funzioni handler dei segnali per i server e il supervisor sfruttano le funzioni di salto incondizionato *sigsetjmp* e *siglongjmp* per trattare i segnali nel loro normale flusso di esecuzione.

2.6 Gestione della concorrenza

Gli unici accessi concorrenti a risorse si verificano nei confronti della pipe unnamed utilizzata da un dato server per comunicare con il supervisor, che viene utilizzata in scrittura da tutti i thread appartenenti al server. L'accesso in regime di mutua esclusione viene realizzato tramite una lock *pthread_mutex_t* associata alla pipe.

3 Osservazioni

Secondo l'algoritmo scelto per affrontare il problema, il secret di un dato client viene stimato correttamente se il client seleziona consecutivamente lo stesso server almeno una volta per l'invio del proprio messaggio. Per l'esecuzione di un client con parametri p, k e w, ciò avverrà con probabilità:

$$\alpha_{p,w} = 1 - \left(\frac{p-1}{p}\right)^{w-1}$$

Per i parametri relativi allo script di test *test.sh*, la probabilità che il secret di un dato client sia stimata correttamente è:

$$\alpha_{5,20} = 0.985$$

Il calcolo della probabilità dei successi totali per un test effettuato con l'esecuzione di n client richiede uno studio più approfondito. Infatti date

$$X_n = \{0, 1\}, n \in \mathbb{N}$$

variabili aleatorie discrete indicatrici per il successo o fallimento della stima del secret dell'n-esimo client, e data una coppia di variabili aleatorie relative a due diversi client, esse sono in generale dipendenti.