Università di Pisa
Dipartimento di Informatica
Corso di Laurea in Informatica

Relazione progetto Laboratorio di Sistemi Operativi

a.a 2018-2019

Pace Antonio, 559397

Sistema di out-of-band signaling

1. Moduli e strutture d'appoggio

Ho diviso il progetto in 5 moduli:

- client.c:
  genera id e secret e comunica con il server.

- supervisor.c:
  crea *k* processi server e le relative pipe per la comunicazione delle stime.

- server.c:
  riceve messaggi dai client e calcola le stime.

- linkedlist.c:
  struttura d'appoggio utilizzata per salvare le stime di ogni client. Contiene i metodi *add, replace, stampa* e *del_list*. Ogni elemento contiene id e stima di un client, oltre al puntatore all'elemento successivo. Il metodo *add* serve ad aggiungere la prima stima, il metodo *replace* a sostituirla con una migliore. *Stampa* serve a stampare la lista, scegliendo tra stderr e stdout tramite l'argomento *control*.

- utility.c:
  contiene i metodi *timeval_subtract* (trovato online) e *min*, d'aiuto per calcolare le stime.¶

2. Funzionamento

2.1 Client

Il client riceve come argomento *p, k* e *w,* dove *p* è il numero di server a cui connettersi tra i *k* disponibili, e *w* è il numero di messaggi da mandare.
Genera casualmente un proprio id e un secret (numero compreso tra 1 e 3000), successivamente si connette ai *p* server tramite socket *AF_UNIX* e manda *w* messaggi (contenenti l'id in esadecimale) scegliendo casualmente uno dei server ad ogni iterazione, aspettando *secret* millisecondi tra un invio e l'altro.

2.2 Supervisor

Il supervisor riceve come argomento *k*, ovvero i server da lanciare. Lancia quindi *k* processi server diversi, con *fork* ed *execl,* mettendosi poi in attesa delle stime (da ricevere tramite pipe anonime) con una *select.* Infine salva tutte le migliori stime ricevute in una linked list.

2.3 Server

I server vengono lanciati dal supervisor, ricevono messaggi (anche ripetuti) da uno o più client. Ha il compito di stimare il *secret* di ogni client da cui riceve messaggi, considerando il tempo passato tra un messaggio e l'altro.

2.4 Misurazione

La misurazione è svolta dal server, calcola il tempo passato tra un messaggio e l'altro tenendo sempre il minimo.

2.5 Segnali

Il server ignora SIGINT e termina con SIGTERM inviato dal supervisor. Nel supervisor c'è *sigint_handler* che gestisce SIGINT, quando arriva stampa il contenuto della lista delle stime sullo *stderr*, se ne arrivano due in meno di un secondo, stampa la lista nello *stdout* e termina.¶

3. Makefile, testing e misura

3.1 Makefile

Il Makefile contiene 3 target:

- all:
  creazione eseguibili.
- clean:
  rimozione file eseguibili, file oggetto e file socket.
- test:
  esegue un test di tutto il codice.

3.2 Run.sh

Semplice script bash che genera il supervisore con argomento 8 e 20 client lanciati 2 alla volta con argomenti *5, 8* e *20*.
Aspetta poi 60 secondi, lanciando un SIGINT ogni 10 secondi, infine lancia un doppio SIGINT che fa terminare il programma, e lancia lo script di misurazione.
L'output di client e supervisor è salvato su client.log, supervisor.log e stderr.log (che contiene lo stderr del supervisor).

3.3 Misura.sh

Lo script di misura analizza il contenuto di client.log e supervisor.log, ricava id e secret dei client e li confronta con le stime stampate dal supervisor, calcola la differenza tra il valore stimato e il valore reale e restituisce il numero di stime corrette (stime che variano non più di 25 unità dal valore reale) e l'errore medio di stima. L'output è salvato su logmisura.log.

3.3 Testing

Per compilare ed eseguire il processo, scompattare il file ed eseguire *make test*. L'output sarà disponibile sui file di log. Al termine non rimuovo i file di log, così da rendere disponibile il controllo di tutto l'output e non solo della misura.
Il progetto è stato testato su MacOs 10.15 Catalina.