Università di Pisa
Dipartimento di Informatica
Corso di Laurea in Informatica

Progetto di Laboratorio di Sistemi Operativi

a.a. 2018-19 (#2)

Introduzione

Lo studente dovrà realizzare un sistema per l'*out-of-band signaling*. Si tratta di una tecnica di comunicazione in cui due entità si scambiano informazioni senza trasmettersele direttamente, ma utilizzando segnalazione collaterale: per esempio, il numero di errori "artificiali", o la lunghezza di un pacchetto pieno di dati inutili, o anche il momento esatto delle comunicazioni.

Nel nostro caso, vogliamo realizzare 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 stia catturando i dati in transito.

Nel nostro progetto avremo *n* client, *k* server, e 1 supervisor. Viene lanciato per primo il supervisor, con un parametro *k* che indica quanti server vogliamo attivare; il supervisor provvederà dunque a creare i *k* server (che dovranno essere processi distinti). Gli *n* client vengono invece lanciati indipendentemente, ciascuno in tempi diversi. Il funzionamento di ciascuno dei componenti è descritto nel seguito.

Il client

All'avvio, il client genera il suo *secret*, che è costituito da un numero fra 1 e 3000, e il suo ID unico, che è un intero a 64 bit; entrambi devono essere generati pseudo-casualmente (si ricordi di inizializzare il generatore casuale in maniera appropriata). Un client riceve sulla riga di comando tre valori interi, p, k e w, con $1 \leq p <$ k e $w > 3p$. Il secondo dovrà essere lo stesso passato al supervisor, che controlla quanti server sono attivi sul sistema.

All'inizio, il client deve stampare sul proprio *stdout* un messaggio nel formato "CLIENT *id* SECRET *secret*", in cui *id* è rappresentato da un numero esadecimale (senza prefissi particolari), e *secret* da un numero decimale. Il client dovrà quindi scegliere casualmente *p* interi distinti compresi fra 1 e *k*, e si collegherà (tramite socket, vedi avanti) ai *p* server corrispondenti agli interi scelti. Inizierà un ciclo in cui, ad ogni iterazione, sceglierà casualmente un server fra i *p* a cui è collegato, gli invierà sulla socket corrispondente un messaggio contenente il proprio ID unico (il formato sarà un buffer di 8 byte contenente l'ID in network byte order), e attenderà *secret* millisecondi prima di ripetere (si può usare la system call **nanosleep(2)** per l'attesa). Dovranno essere inviati complessivamente *w* messaggi (in totale, non per ciascun server). Una volta completato il proprio compito, il client termina, dopo aver stampato un messaggio "CLIENT *id* DONE".

Il server

Un server (che viene lanciato dal supervisor) per prima cosa apre una socket nel dominio AF_UNIX (si usi come indirizzo la stringa "OOB-server-*i*", in cui *i* è il suo numero progressivo fra 1 e *k*). Procede quindi ad attendere connessioni da parte dei client, che come detto possono arrivare in qualunque momento. Per ogni connessione, il server dovrà osservare il momento di arrivo dei messaggi da parte del client, e in particolare il tempo trascorso fra messaggi consecutivi. Per come è strutturato il client, il tempo fra due messaggi consecutivi allo stesso server sarà un multiplo del suo *secret.* Compito del server è di stimare il *secret* di ogni client, e – nel momento in cui la connessione viene chiusa da quel client – informare il supervisor di qual'è la sua migliore stima, inviando un messaggio che contenga l'ID del client e la sua stima del *secret*.

All'avvio il server deve stampare sul suo stdout un messaggio nel formato "SERVER *i* ACTIVE". Per ogni nuova connessione ricevuta, deve stampare un messaggio "SERVER *i* CONNECT FROM CLIENT". Per ogni messaggio ricevuto sulla socket, deve stampare su stdout un messaggio "SERVER *i* INCOMING FROM *id* @ *t"*, in cui *id* è l'ID del client che ha inviato il messaggio, e *t* è il tempo di arrivo (espresso come un numero, con unità di misura a discrezione). Per ogni chiusura della socket, deve stampare su stdout un messaggio "SERVER *i* CLOSING *id* ESTIMATE *s"*, in cui *s* è il valore stimato da questo client per il *secret*

del client *id*.

Al momento della chiusura, il server deve comunicare al supervisor la sua migliore stima per il valore del *secret* di *id* (la stessa stampata nel messaggio). La comunicazione fra server e supervisor è realizzata tramite pipe anonime come descritto sotto.

Il supervisor

All'avvio, il supervisor dovrà stampare sul suo *stdout* un messaggio "SUPERVISOR STARTING *k*", in cui *k* è il numero di server da lanciare (che il supervisor riceve come argomento sulla riga di comando). A questo punto, dovrà lanciare (come processi distinti) i *k* server, e mantenere con ciascun server una connessione tramite pipe anonime con cui potrà ricevere le stime dei *secret* dei client da parte dei server. Per ogni nuova stima ricevuta, il supervisor dovrà stampare su *stdout* un messaggio "SUPERVISOR ESTIMATE *s* FOR *id* FROM *i*", in cui *s* è il valore della stima del *secret*, *id* è l'ID del client a cui si riferisca la stima, e *i* è l'indice del server che ha fornito la stima.

Il supervisor riceverà ovviamente stime da più server per lo stesso client. Deve quindi utilizzare le diverse stime arrivate fino a un dato momento per determinare il valore più probabile del *secret* "vero"; chiamiamo questo valore S_{id}. Quando il supervisor riceve un segnale SIGINT, deve stampare sul suo *stderr* una tabella che mostri le sue stime correnti, in cui ciascuna riga ha il formato "SUPERVISOR ESTIMATE S_{id} FOR *id* BASED ON *n*" – in cui *n* rappresenta il numero di server che hanno fornito stime per *i d* fino a quel momento. Se invece riceve un "doppio Ctrl-C", ovvero un secondo segnale SIGINT a distanza di meno di un secondo dal precedente, il supervisor deve terminare, stampando prima la stessa tabella su *stdout* seguita dal messaggio "SUPERVISOR EXITING".

Misurazione

Si dovrà realizzare uno script bash di nome *misura* che, ricevuti come argomenti i nomi di un insieme di file contenenti l'output di supervisor, server e client, ne legga e analizzi i contenuti, e stampi delle statistiche su quanti secret sono stati correttamente stimati dal supervisor (intendendo per stima corretta un *secret* stimato con errore entro 25 unità rispetto al valore del *secret* vero), e sull'errore medio di stima.

Makefile

Il progetto dovrà includere un makefile avente, fra gli altri, i target *all* (per generare tutti gli eseguibili), *clean* (per ripulire la directory di lavoro dai file generati), e *test*. Quest'ultimo deve eseguire un ciclo completo di test, lanciando il supervisor con 8 server, e dopo un'attesa di 2 secondi dovrà lanciare un totale di 20 client, ciascuno con parametri 5 8 20. I client andranno lanciati a coppie, e con un attesa di 1 secondo fra ogni coppia.

Dopo aver lanciato l'ultima coppia, il test dovrà attendere 60 secondi, inviando nel frattempo un SIGINT al supervisor ogni 10 secondi; al termine dovrà inviare un doppio SIGINT, e lanciare lo script *misura* sui risultati raccolti (tramite redirezione dello stdout dei vari componenti su appositi file).

Note finali

Nel realizzare il sistema oggetto del progetto, si tenga presente che il protocollo di comunicazione fra supervisor e server può essere privato, mentre quello fra server e client è pubblico. In altre parole, il vostro sistema supervisor + server deve funzionare anche se a connettersi sono client diversi da quelli sviluppati da voi. Analogamente, i vostri client dovranno essere in grado di interoperare con supervisor+server sviluppati da altri.

Ci si attende che tutto il codice sviluppato sia, per quanto possibile, conforme POSIX. Eventuali eccezioni vanno documentate nella relazione di accompagnamento. La consegna dovrà avvenire, entro il termine pubblicato, attraverso l'invio per posta elettronica al vostro docente di un archivio con nome *nome_cognome*.tar.gz contenente tutto il necessario. Scompattando l'archivio in una directory vuota, dovrà essere possibile eseguire i comandi *make* (con target di default) per costruire tutti gli eseguibili, e *make test* per eseguire il test e vederne i risultati. L'archivio dovrà anche contenere una breve relazione (3-5 pagine) in formato PDF in cui illustrate il vostro progetto.