Corso di Sistemi Operativi e Laboratorio

A.A. 2019/2020

Relazione progetto:

Out-of-Band Signaling

Matteo Stefanelli

543781

Introduzione

Il progetto sviluppato consiste in un sistema distribuito, dove il client comunica il suo secret a un supervisor, mandando vari messaggi a più server, in modo che la comunicazione risulti sicura.

Il progetto è composto da:

- Sorgenti
  - supervisor.c
  - server.c
  - client.c
- Script:
  - test.sh
  - misura.sh

Qui di seguito verranno esposti la struttura dei programmi nelle singole parti e infine il funzionamento degli script.

Struttura dei programmi

Supervisor

Il supervisor è composto da un singolo flusso di esecuzione, dapprima verifica l'inserimento del parametro *k*, per poi inizializzare il gestore del *SIGINT*, di cui parleremo in seguito.

A seguito della creazione di *k* pipe anonime distinte (una per ogni server), entra in un ciclo dove crea i *k* server, attraverso la funzione *fork*, salvando i pids dei figli nell'array *pids* (questi serviranno dopo per la terminazione). In questo ciclo i figli chiuderanno i descrittori delle pipe su cui non devono scrivere, ovvero tutti tranne l'i'esimo e poi verranno lanciati attraverso la *exec* i *k* server.

A questo punto il supervisor entrerà in un loop, in ascolto su tutte le k pipe attraverso la funzione *select*. Per come è stato strutturato il server (vedi in seguito), il supervisor dovrà fare due *read* consecutive sul descrittore della pipe pronta a scrivere: la prima avrà il compito di leggere l'id del client, passato in forma decimale a 64 bit, e la seconda dovrà leggere un intero che rappresenterà la migliore stima per quel client da parte del server scrivente. Questi saranno poi passati alla funzione *inserEl* che ha il compito di verificare l'esistenza dell'id in un array ed eventualmente aggiungere il nuovo id con la sua stima o aggiornare la stima ricevuta qualora essa sia valida. In particolare, una stima viene ritenuta valida se è maggiore di zero e se è minore rispetto alla stima correntemente salvata per quell'id, se presente.

Al momento della ricezione del *SIGINT,* il *gestore* ha il compito di verificare l'intervallo di tempo tra due segnali *SIGINT* ricevuti consecutivamente e agisce come segue:

a. il *SIGINT* preso in considerazione è il primo segnale ricevuto
   1. prende il tempo di ricezione e inizializza la variabile *tint (Tempo INTerruzione)*
   2. chiama la funzione *dati_clients* per la stampa dei dati raccolti fino a quel momento su *stderr*
b. il SIGINT preso in considerazione non è il primo segnale ricevuto
   1. verifica che l'intervallo di tempo tra questo segnale e il precedente non sia superiore a 1 secondo

i. se lo è, stampa i dati raccolti su *stderr*
ii. se non lo è, stampa i dati raccolti su *stdout,* manda un segnale *SIGUSR1* a tutti i figli memorizzati nell'array *pids* e infine chiama il *cleanup* di tutte le variabili allocate fino a quel momento e termina

La funzione di *cleanup* è stata gestita tramite il supporto di un'altra funzione, ovvero *cleanup_push,* che ha il compito di passare l'indirizzo di tutte le variabili allocate nello heap, memorizzandoli in un array di appoggio. La funzione di *cleanup* effettuerà quindi la free su ogni elemento dell'array per poi liberare anche la memoria allocata per l'array stesso.

Server

Il server è composto da più flussi di esecuzione. Inizialmente il main thread provvederà alla creazione di una socket su cui rimarrà in ascolto fino all'arrivo del segnale *SIGUSR1* da parte del supervisor. Ad ogni rischiesta di connessione, è associato un nuovo thread che si occuperà esclusivamente di gestire quel client.

Inizierà un ciclo in cui il thead rimane in ascolto sulla connessione aperta, misurando il tempo tra due messaggi consecutivi e tenendo nella variabile *secret* solo la migliore stima rilevata fin'ora. Il thread infatti terrà il minimo intervallo tra due messaggi consecutivi ricevuti, e quella rappresenterà la migliore stima. Quando il thread riceverà un EOF da parte del client (ovvero quando la *read* restituirà 0) uscirà dal ciclo e, attraverso il meccanismo del lock, si metterà in attesa di poter scrivere la sua miglior stima al supervisor. Se il server ha ricevuto il segnale da parte del supervisor, esce dal ciclo, mentre se è ancora in ascolto, terminerà rilasciando la memoria occupata.

Il meccanismo delle lock per scrivere sulla pipe è stato adottato affinché non ci potesse essere concorrenza tra i vari thread dello stesso processo server, in quanto vengono effettuate due *write* consecutive, una per l'id del client e l'altra per la relativa stima del secret, al fine di non rallentare l'operato della *select* attraverso varie operazioni sulla stringa contenete entrambi i dati. Inoltre, i thread si sospendono sulla lock, nel caso fosse occupata la pipe, solo prima di terminare, evitando così rallentamenti di operazioni importanti.

La funzione *gestore* del server, che gestisce la *SIGUSR1*, ha il solo compito di settare una variabile condivisa per l'intero processo che viene impostata a zero quando il segnale viene ricevuto. La memoria occupata, viene quindi liberata, vengono chiusi i descrittori di connessioni con eventuali clients e quello della pipe anonima verso il supervisor.

Client

Dopo una prima fase di controllo dei parametri di ingresso, il client provvede a installare un gestore per il segnale di *SIGPIPE*, per la terminazione "pulita" del programma. Successivamente provvederà a generare prima il secret usando un classico generatore standard e in seguito il suo id a 64 bit attraverso la composizione di due numeri a 32 bit generati casualmente usando un altro seme e un altro generatore.

Dopo una prima fase di connessione a *p* server casuali, il client genera un numero casuale compreso tra *0* e *p-1* (inclusi) che verrà usato come indice per l'array contente i file descriptor delle connessioni

aperte in precedenza e mandando il suo id convertito in network byte order al server selezionato. Ripeterà questo ciclo per *w* volte prima di chiudere le connessioni e liberare la memoria attraverso la funzione di *cleanup* (la stessa utilizzata dal supervisor).

Test.sh e Misura.sh

Lo script *test.sh* provvede a lanciare come da consegna, 20 clients e 8 servers, stampando su terminale lo *stderr* di supervisor, in modo da poter vedere la stampa delle stime effettuata dal gestore sullo *stderr*, mentre direziona lo *stdout* di server+supervisor sul file *supervisor.log* e quello del client su *client.log*. Viene poi lanciato lo script di misura con parametri numero clients (20) log stdout supervisor+server e log stdout client.

A questo punto parte lo script di misura che utilizza il linguaggio *awk* per estrarre i dati dai file di log creando due file dove lavorerà in seguito. In questi due file vengono inseriti per ordine di id, in modo da poter vedere ad occhio se sono unici, solo id e stima in *Supervisor.log* e id e secret in *Client.log*.

Parte poi un ciclo dove per ogni riga di *Supervisor.log* viene cercato il match in *Client.log* e viene calcolata la differenza tra il secret e la stima. Infine, viene stampata una tabella riepilogativa.