Relazione finale progetto SOL

autore: Antonio Carta

I file

la directory principale contiene tutti i sorgenti ed il makefile. La cartella /test contiene i test mentre la cartella /build contiene i file compilati.
Per compilare dare il comando make dalla directory principale.
make test esegue il test definito nella specifica del progetto.
Un header, common.h, contiene varie definizioni utili come la gestione degli errori o la conversione da network a host byte order per interi a 64 bit.
AVL.c implementa gli alberi binari di ricerca, usati dal supervisor per salvare le stime.

Client

La generazione del seme della rand viene effettuata tramite il tempo in microsecondi per poter avviare più client nello stesso secondo e garantire che abbiano tutti un seme univoco.
La scelta dei server invece viene effettuata mischiando un array (O(p)) contenente tutti i possibili indici dei server e sceglie come server quelli che hanno l'id nelle prime p posizioni.
Per garantire che l'id creato abbia 64 bit viene usato l'header inttypes.h. Se questo non fosse disponibile sarebbe necessario definire nel file common.h la typedef di uint64_t e la macro PRIx64, usata come formato per la printf.

Server

il server riceve 2 argomenti: il proprio id e il file descriptor in cui si trova la pipe che è stata inizializzata dal supervisor e verrà usata per la comunicazione server-supervisor.
Dopo aver inizializzato il socket attende le connessioni da parte dei client. Per ogni connessione crea un nuovo thread che si occuperà di ricevere i messaggi da quel particolare client, ricavarne una stima ed inviarla al supervisor tramite la pipe.
La versione multithread rispetto a quella con la select permette una gestione più semplice delle varie stime. Con la select avremmo dovuto salvare tutte le stime in una struttura dati complessa, dato che queste venivano calcolate tutte dallo stesso server. In questo modo invece ogni thread calcola una sola stima.
La stima viene effettuata calcolando tutti gli intervalli di tempo tra messaggi consecutivi e considerando come stima il minimo, dato che questi saranno sicuramente un multiplo del secret. Se riceve un solo messaggio quindi non ha una stima e non invia niente al supervisor (w>3p garantisce che il supervisor riceverà comunque delle stime). Se invece la stima è maggiore di 3000, il server trova il minimo intero per cui può dividere la stima e farla diventare minore di 3000, in maniera tale da esser certi che la stima sia sempre minore di 3000 e sia un valore congruente con le altre stime ricevute. Il messaggio inviato al server è composto da: stima, id del client, id del server (la pipe è condivisa con tutti i server), e tempo di arrivo dell'ultimo messaggio

Supervisor

il supervisor prima di tutto prepara i signal handler di SIGINT e SIGALRM. Entrambi si limitano a settare delle variabili globali di tipo sig_atomic_t. In questo modo la gestione del segnale è minima. Queste variabili verranno controllate successivamente dal loop principale del programma in cui verrà gestita la stampa e la terminazione. Il segnale SIGALRM serve per avvisare che è passato un secondo dall'ultimo SIGINT.
In seguito il supervisor crea la pipe con cui comunicherà con i server e li crea, salvandosi tutti i pid in un array. Questi vengono usati all'uscita per terminare i server.
Nel ciclo principale viene effettuata la lettura della pipe, viene aggiornato l'albero (binario di ricerca) delle stime e vengono gestite le eventuali operazioni richieste dai segnali.

stima
Ogni server invia al supervisor un messaggio contenente la propria stima, il proprio id, l'id del client e il tempo di arrivo dell'ultimo messaggio ricevuto dal client. Il supervisor invece salva nell'albero delle stime l'id del client, la stima, il numero di server che hanno effettuato la stima e un tempo. Quando il supervisor riceve un nuovo messaggio attraverso la pipe aggiorna l'albero delle stime. Se il supervisor non ha ancora stime per quel client, la stima ricevuta diventa la sua stima, altrimenti aggiorna il numero di server che gli hanno inviato la stima per quel client come tempo prende il maggiore tra quello suo e quello ricevuto e come stima prende il minore tra la sua stima precedente, la stima del server e l'intervallo di tempo tra il suo tempo e quello del server.
La scelta della stima minima dei server è abbastanza ovvia dato che ogni stima è multiplo del secret, e permette di raggiungere una buona percentuale di stime corrette (facilmente calcolabile una volta dati p e w), se trascuriamo la probabilità che gli intervalli calcolati siano sbagliati di più di 25 ms rispetto a quelli reali. Ma anche l'intervallo tra l'ultimo messaggio ricevuto da 2 server diversi è un multiplo del secret e può essere usato per la stima. Se consideriamo gli ultimi due messaggi inviati dal client abbiamo due possibilità: i messaggi sono inviati allo stesso server (che in questo caso indovinerà la stima) oppure sono inviati a due server distinti (e quindi l'intervallo tra i due messaggi sarà il secret).

misura
il file misura sfrutta gli array associativi della bash per salvare i secret e le stime, ricavate dai file che riceve per argomento. I file vengono letti tramite la read e la redirezione dello standard input. Infine calcola e stampa le statistiche e tutti i secret che sono stati sbagliati.

makefile
come richiesto nella specifica il makefile contiene i target all, test e clean. Il makefile si trova nella directory principale del progetto e permette di compilare tutti i file che devono essere ricompilati ed effettuare il test.