Relazione finale sul progetto di Sistemi Operativi

OOB: Out of Band signaling

Marco Antonio Corallo, 531466

04 Novembre 2019

Introduzione

Il modulo di *Laboratorio di Sistemi Operativi* prevede lo svolgimento di un progetto da sviluppare utilizzando il linguaggio C in ambiente Linux.
Il progetto deve essere svolto singolarmente da un solo studente.
Questo documento descrive la struttura complessiva del progetto chiamato Out-of-Band signaling.
La scadenza per la consegna del progetto coincide con la scadenza dell'appello straordinario (Novembre 2019) dell'a.a 2018/2019.
Il progetto dovrà essere sviluppato e documentato utilizzando gli strumenti, le tecniche e le convenzioni presentate durante il corso.
L'obiettivo del progetto è realizzare un sistema client-server in cui viene comunicato un codice segreto da parte del client senza però trasmetterlo.
Lo scopo è rendere difficile intercettare il secret a chi stia catturando i dati in transito.

Moduli

Il progetto, come già detto, consiste in un sistema *client-server*, in cui si possono avere un numero arbitrario di clients che comunicano con un numero altrettanto arbitrario di servers.
La gestione dei servers viene demandata ad un modulo *supervisor.*

Supervisor

Il supervisor viene avviato con un argomento *k*, che rappresenta il numero di servers da avviare e gestire.

Se questo argomento non viene passato all'avvio, può comunque essere inserito immediatamente dopo.

Avviati i *k* servers, viene stabilita una pipe anonima con ognuno di questi, memorizzando ogni coppia di *file descriptors* in un array bidimensionale di dimensione $k \times 2$.

La scelta che è stata fatta è di utilizzare *read non-bloccanti*.

I messaggi da parte dei servers sono infatti gestiti tramite *Select* all'interno di un ciclo nel main thread.

Il ciclo verrà interrotto dal gestore di segnali solo nel momento in cui vengono mandati due segnali di interruzione consecutivamente.

Ogni messaggio contenente una stima da parte dei servers, viene salvato in una struttura di tipo *estimate_t*, il cui funzionamento è spiegato nella sezione dedicata alle strutture dati, a sua volta salvata in una *tabella hash* che utilizza l'***ID*** del client come ***chiave***, e la struttura ***estimate_t*** come ***valore*** di ogni elemento.

Server

L'impostazione data al modulo server vede il main thread rivestire il ruolo di *dispatcher* di connessioni da parte dei clients, utilizzando la *select* per controllare che la socket utilizzata per le connessioni sia pronta.

All'arrivo di una richiesta di connessione viene lanciato un *thread* che si occuperà di quel client per tutta la durata della connessione.

Questo viene fatto tramite una struttura dati che ho chiamato *workerMask*, la quale *segnala* anche la terminazione dell'esecuzione del thread.

Una volta che il thread finisce il suo compito, infatti, il main thread si occupa di effettuare la *join* del thread all'interno dello stesso ciclo di dispatch.

Dal ciclo è possibile uscire solo con l'arrivo di una *SIGPIPE* da parte del supervisor.

Client

Il client genera *ID* e *Secret* in maniera pseudo-casuale, utilizzando come seme di inizializzazione del generatore pseudo-casuale un'elaborazione sui valori restituiti dalle funzioni *clock()*, *time()* e *getpid()*, in modo di ridurre al minimo la possibilità di avere ID uguali per clients diversi.

Il server a cui connettersi ad ogni iterazione è stato scelto tra i *p* servers pre-selezionati prima ancora di entrare nel ciclo, memorizzando le *w* scelte in un array. Questa scelta è stata fatta al fine di ridurre il numero di operazioni – e quindi il tempo trascorso – tra l'invio di un messaggio e il successivo.

In questo modo la stima fatta dal server può essere più precisa, sopratutto quando il sistema giri su macchine con tempi di elaborazione più alti.

Strutture Dati

Tabella Hash

L'implementazione utilizzata è la *icl_hash* fornita dai docenti del corso.

La tabella è stata utilizzata all'interno del Supervisor per memorizzare le informazioni sui clients e le rispettive stime.

Come chiave di ogni elemento è stato utilizzato l'**ID** del client, memorizzando come **valore** una struttura **estimate_t** definita nel rispettivo file header.

Estimate

Struttura dati utilizzata per memorizzare le informazioni dei clients all'interno della tabella hash.

L'idea alla base è mantenere per ogni client un'associazione *<ID,StimaMigliore>*.

La struttura memorizza infatti: l'ID del client, un valore aggiornato all'ultima stima calcolata da un server ed il numero di servers che hanno fornito un valore per quel client.

WorkerMask

Si tratta di una struttura dati utilizzata dal modulo Server per tenere una traccia del client gestito da ogni thread.

```
typedef struct workerMask{
    int         fd;         //FD comunicazione con un client
    pthread_t   tid;        //ID del thread che se ne occupa
    int         end;        //Flag di chiusura FD
    int         attivo;     //Ancora in uso
} workerMask_t;
```

La struttura mantiene il FD tramite il quale il server comunica con il client, l'ID del thread che se ne occupa, un flag ad indicare se il thread attende di essere terminato ed un ultimo flag ad indicare se quel file descriptor è in uso.

In questo modo è possibile utilizzare un array contenente le informazioni sui clients in esecuzione, liberando lo spazio non appena questi terminano, riservandolo ai prossimi clients che si connetteranno.

Approximate

L'algoritmo scelto per valutare la stima di un secret è una leggera variante del MCD.

Se uno tra *a* e *b* è uguale ad 1, il valore restituito dall'algoritmo sarà l'altro numero.

Se l'MCD tra i due è uguale ad 1, l'algoritmo restituirà il minimo tra *a* e *b*.

Questa leggera modifica serve a prevenire i casi in cui *a* o *b* sono stime errate per poche unità accumulate dal ritardo di propagazione.

Timeval_subtract & mix

Si tratta delle funzioni utilizzate rispettivamente per calcolare la differenza tra due orari in millisecondi e per inizializzare il generatore pseudo-casuale con un grado di casualità maggiore dei metodi più tradizionali.

Entrambe le funzioni sono state trovate sul web. Cito e ringrazio:

1. https://www.linuxquestions.org/questions/programming-9/how-to-calculate-time-difference-in-milliseconds-in-c-c-711096/
2. https://stackoverflow.com/a/323302

Testing e misurazione

Per lo script di testing si è scelto di reindirizzare stdout e stderr di ogni entità in esecuzione verso un diverso file di testo.

Questi files verranno passati come argomenti allo script di misurazione, dopodichè verranno eliminati automaticamente.