18/02/2021

il progettino

Reti informatiche 2020-2021

Marco Lucio Mangiacapre
UNIPI – INGEGNERIA INFORMARICA

Il progettino

Questo breve documento fornisce una spiegazione estremamente sintetica delle scelte implementative de “il progettino”, della filosofia che v’è dietro e dell’organizzazione del processo di sviluppo.

Cardini filosofici

Lo sviluppo di questo mio progetto è stato fortemente influenzato e ha portato a rafforzare queste mie convinzioni e dai seguenti concetti:

- Il codice deve esse modulare e riutilizzabile.
- Incapsulamento del codice: le strutture dati si manipolano esclusivamente tramite le funzioni predisposte, mai direttamente.
- I puntatori a void sono un male necessario: serve generalità.
- Il codice si testa e si debugga man mano: tutto, anche quello dei test.
- Thread diversi per compiti diversi.
- Il thread principale è l’unico ad avere il diritto di prendere degli input.
- Mezzo milione di caratteri sono sufficienti.

Struttura dei programmi

Sia peer che server sono processi multi thread, questo perché ho ritenuto opportuno separare nettamente il codice che gestisce l’interazione con l’utente da quello che gestisce lo scambio di messaggi attraverso la rete.

Struttura del peer:

- Thread principale: gestisce l’input dell’utente e regola l’utilizzo del terminale;
- Thread “ENTRY”: avviato per primo, gestisce i registri posseduti dal peer e l’aggiunta di un nuovo registro alle 18:00;
- Thread “UDP”: avviato dal main thread subito dopo del thread “ENTRY” gestisce la ricezione e l’invio dei messaggi UDP, di fatto si occupandosi quindi della comunicazione con il server;
- Thread “TCP”: avviato dal main thread non appena il peer riesce a connettersi al server, si occupa di gestire l’interazione tra un peer e i suoi vicini.

Struttura del server:

- Thread principale: gestisce l’input dell’utente e regola l’utilizzo del terminale;
- Thread “UDP”: avviato dal main thread all’inizio, si occupa dell’interazione con i peer.

Formato dei messaggi

Tutti i messaggi che si scambiano server e peer sono in formato binario, indipendentemente dal fatto che siano inviati mediante socket TCP oppure UDP. I messaggi posseggono tutti un’intestazione di 8 byte così strutturata:

- 2 bye “sentinella”: obbligatoriamente a 0, introdotti per permettere di riconoscere con facilità ed eventualmente gestire in una versione futura del progetto messaggi testuali a scopo di debugging. Un messaggio con questi due byte a 0 si dice regolare.
- 2 byte che indicano la tipologia del messaggio: supposto di avere un messaggio regolare questi due byte permettono di riconoscere il tipo del messaggio per gestirlo opportunamente.
- 4 byte per uno “pseudo ID”: inutilizzati nella maggior parte dei casi, permettono quando utilizzati di riconoscere immediatamente messaggi duplicati e di associare messaggi di richiesta e risposta. Un esempio di utilizzo sono le richieste di boot dei peer inviate al server, situazione in cui se non venissero riconosciuti i messaggi duplicati si rischierebbe di portare la rete in uno stato inconsistente.

Il corpo dei vari tipi di messaggi validi può avere dimensione fissa o variabile, nel qual caso è sempre presente un campo che ne specifica la lunghezza. Vedere “messages.h”.

Utilizzo

Per avviare il tutto assicurarsi di avere installato tmux (“sudo apt-get install tmux”) sulla macchina dove si esegue il testing, accedere alla cartella del progetto e digitare il comando make senza argomenti. Automaticamente sarà avviata la compilazione dei sorgenti, la generazione degli eseguibili, la generazione dei registri sui quali eseguire i test e il lancio di un server e di cinque peer in una stessa finestra di una sessione tmux. Vedere in basso per una brevissima (e assolutamente insufficiente) introduzione a tmux.

Funzionamento

L’interazione con il server è banale e segue le specifiche, qui fornirò solo una brevissima descrizione dell’interazione tra peer.

Connessione

È compito dei peer che per ultimi si sono connessi alla rete avviare la procedura di connessione con i peer già connessi. Il protocollo di connessione prevede l’invio di un messaggio di richiesta di connessione e uno di risposta.

FLOODING

Il protocollo per il flooding delle entry è complicato e sarò felice di descriverlo in sede di orale, l’idea alla base è tuttavia semplice: eseguire una “deep-first search” asincrona e distribuita. Più istanze del protocollo possono eseguire contemporaneamente.

Si poteva fare di meglio?

Sì, ci sono numerose ottimizzazioni che si potrebbero apportare al progetto. Una tra tutte: avrei potuto implementare un meccanismo di “heartbeat” in grado di rilevare automaticamente il crash del server o dei peer e un efficace protocollo di ripristino della rete in seguito alla terminazione anomala di un numero arbitrario di membri. Tuttavia, non ho ritenuto necessario implementare tutto ciò per quello che dovrebbe essere un semplice progetto per un esame sebbene sia pronto in sede di orale a discutere di come avrei potuto apportare le modifiche di cui ho accennato in questo paragrafo.

TMUX

Segue una breve introduzione a tmux e ai comandi utili per testare il progetto.

Di cosa si tratta?

Tmux, “terminal multiplexer”, è un utilissimo programma per permettere di partizionare un terminale per veder eseguire più programmi all’interno della stessa finestra. Si veda “man tmux” per una descrizione esaustiva delle sue funzionalità.

Per testare il mio lavoro è sufficiente conoscere questi comandi:

- Dall’interno di una sessione tmux:
  - CRTL+B FRECCIA DIREZIONALE: per selezionare il “pannello” di interesse;
  - CRTL+B CRTL+Z: per disconnettersi dalla sessione lasciando i vari processi a funzionare in background;
  - CRTL+B “:kill-session”: per terminare tutti i processi attivi nella sessione corrente e tornare al terminale normale. Fare attenzione ai due punti ‘:’.
  - CTRL+B PAG-SU: per poter scorrere verso l’alto la console, premere q per tornare allo stato iniziale.
- Da un nomale terminale:
  - tmux a: per accedere all’ultima sessione da cui ci si è disconnessi

Nota: una sessione viene automaticamente chiusa come termina l’ultimo dei processi che la stava utilizzando.

Per qualsiasi chiarimento sono a disposizione.