Università di Pisa

Dipartimento di Informatica

Relazione del progetto `TURING`

RCL a.a. 18-19

Autore: Michele Zoncheddu

Marzo 2019

Indice

1 Cosa è richiesto nella relazione

- una descrizione generale dell'architettura complessiva del sistema, in cui sono motivate le scelte di progetto;
- uno schema generale dei threads attivati da ogni componente e delle strutture dati utilizzate, con particolare riferimento al controllo della concorrenza;
- una descrizione sintetica delle classi definite ed indicazioni precise sulle modalita di esecuzione.
- una sezione di istruzioni su come compilare ed eseguire il progetto (librerie esterne usate, argomenti da passare al codice, sintassi dei comandi per eseguire le varie operazioni...). Questa sezione deve essere un manuale di istruzioni semplice e chiaro per gli utilizzatori del sistema.

2 Scelte progettuali

Di seguito verranno illustrate le principali scelte progettuali effettuate durante la realizzazione del progetto.

2.1 Architettura

L'architettura del server è stata scelta tra due soluzioni, che sono state trattate più in dettaglio durante il Laboratorio di Reti di Calcolatori:

1. multithread sincrona con I/O bloccante (`Sockets` di `Java IO`);
2. monothread sincrona con I/O non bloccante (`Selectors` di `Java NIO`).

Le due soluzioni hanno pregi e difetti complementari, ma il trade-off principale è tra **velocità** e **scalabilità**. Ai fini di questo progetto didattico è stata ritenuta più importante la reattività, garantita (sotto carichi non eccessivi) dalla prima soluzione.

Per ogni client, il server riserva un thread, che si occuperà di gestire la connessione TCP per l'intero tempo di vita del client.

3 Gestione della concorrenza

I thread sono gestiti da un **cached thread pool**, scelto per la sua elasticità nella gestione delle risorse; sono inoltre indipendenti ed isolati, e per ogni richiesta esterna devono utilizzare uno dei seguenti manager:

- `userManager`: gestisce concorrentemente gli utenti registrati;
- `documentManager`: gestisce concorrentemente i documenti salvati;
- `addressManager`: gestisce concorrentemente gli indirizzi IP multicast da assegnare alle chat dei gruppi di lavoro.

3.1 `userManager`

È il nucleo del servizio di autenticazione di `TURING`, ed è implementato da una tabella hash concorrente (chiave: username).

Viene utilizzato dal server come manager locale per il login e per i controlli di sicurezza delle operazioni; inoltre viene utilizzato dai client come manager remoto per la registrazione, invocando i metodi esposti da una API.

3.2 `documentManager`

Implementato da una tabella hash non concorrente[1] (chiave: username concatenato con il nome del documento), gestisce la creazione, la modifica e l'insieme degli utenti autorizzati alla modifica.

3.3 `addressManager`

Per assegnare e rilasciare su richiesta gli indirizzi multicast per le chat dei documenti, ho utilizzato un *TreeSet* con un comparatore apposito per indirizzi IP, in quanto la classe *InetAddress* non è nativamente comparabile in Java.

Viene riservato un indirizzo da 239.0.0.0 a 239.255.255.255[2] non appena un utente inizia a modificare una sezione di un documento, e liberato quando l'ultimo utente termina la modifica del documento.

[1] In questo caso la corretta gestione della concorrenza non è garantita da una struttura dati concorrente, ma dalle locks delle sezioni, in quanto hanno una granularità più fine.

[2] Organization-Local Scope. Fonte: IANA.