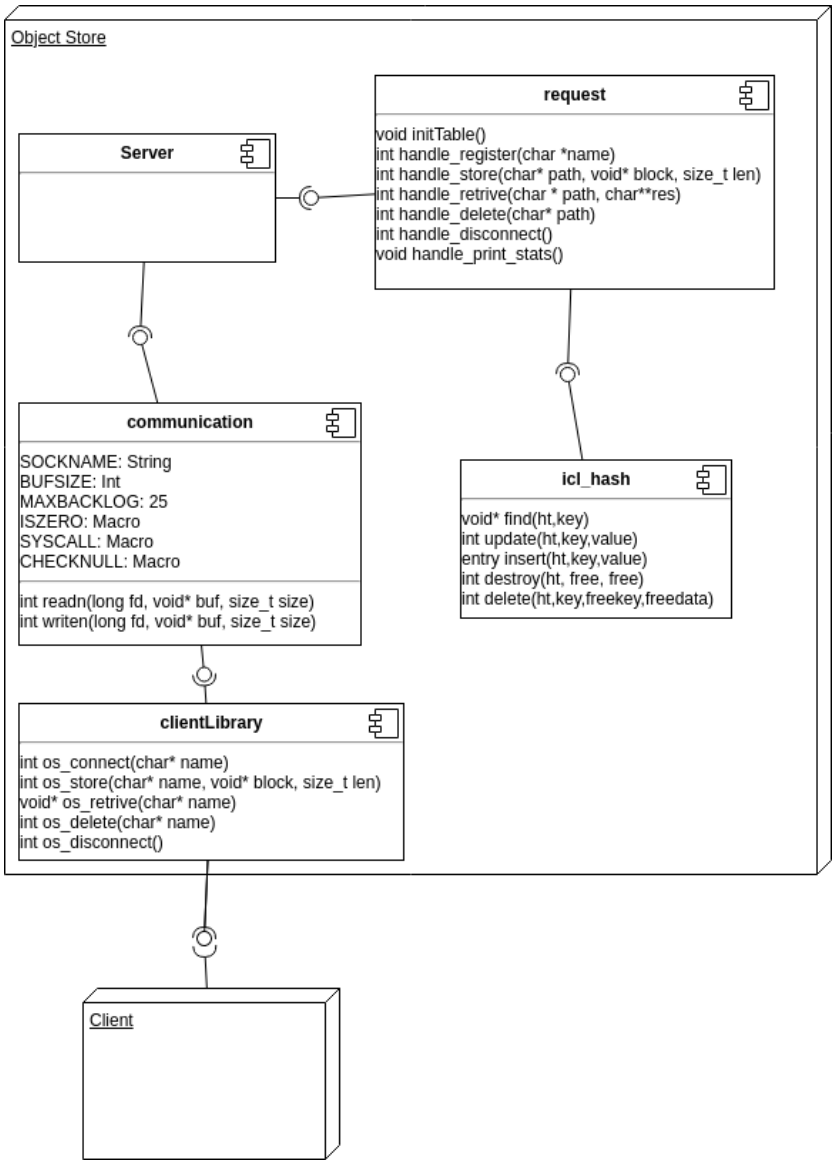
Relazione SOL A.A 2018/19
Gennaro Baratta, corso B matr. 517339

Struttura codice



I codice del server è strutturato nel seguente modo:

Server: inizializza il socket, avvia il thread per gestire i segnali, accetta le nuove connessioni e avvia i thread per gestirle

Request: gestisce la tabella dei client connessi, effettua letture e scritture sul disco, stampa le statistiche dell'object store

Icl_hash: codice fornito dai docenti, ampliato per gestire la concorrenza

Communication: contiene primitive per la comunicazione e macro generiche

ClientLibrary: libreria cliente richiesta dalle specifiche del progetto

Scelte implementative

Si è scelto di utilizzare per l'I/O il paradigma bloccante in multi-threading. Ogni thread creato dal server è dedicato ad un singolo client e c'è un limite massimo di thread attivi contemporaneamente per impedire un dispendio di risorse eccessivo per la macchina che ospita il server. Inoltre, per prevenire la starvation, il server specifica per ogni connessione un timeout massimo per la ricezione tramite l'opzione SO_RCVTIMEO.

Concorrenza della tabella hash

Per evitare un eccessivo consumo di memoria creando un mutex per ogni lista a trabocco, ogni mutex gestisce più di una lista. In aggiunta le mutex sono di tipo read-write per permettere a più thread di leggere il contenuto di memoria ma impedire a più di modificarlo.
Un altro singolo mutex permette la modifica concorrente del campo che memorizza la dimensione della tabella

Gestione dei segnali

I segnali vengono gestiti da un thread diverso la quello principale. In caso di segnali di terminazione viene settato un apposito flag condiviso fra i vari thread che permette di lasciare l'object store in una situazione consistente. Una volta atteso che tutti i thread che gestiscono i client siano terminati, il signal thread effettua un shutdown sul listenfd che permette lo sblocco del server e quindi la sua terminazione.