## Architettura del server

Il server utilizza in totale 3 threads + tanti worker threads quanti specificato nel file di configurazione (almeno uno). I threads sono specializzati in questo modo:

* Master thread: ascolta sul server socket specificato nel file di configurazione, gestisce le nuove connessioni in entrata e avverte i worker threads se i client hanno inviato una nuova richiesta. Si occupa di mantenere una lista di client connessi e fa da fornitore per la coda di client da servire.
* Signal handler thread: gestisce i segnali SIGINT, SIGQUIT, e SIGHUP come da specifica. Se viene ricevuto uno di questi segnali, il thread invia un messaggio al master server, il quale provvede a svegliare i worker e a chiudere i file descriptor dei client connessi, dunque a terminare stampando le statistiche.
* Logging thread: si occupa di mantenere il log del server. I vari thread inviano messaggi tramite un pipe al thread di logging, e questo stampa su stdout e scrive sul file di log.
* Worker threads: si occupano di servire le richieste dei client. Quando il master thread riconosce che il file descriptor di un client è pronto ad una lettura, mette il file descriptor in una coda di client pronti e sveglia un worker. Questo quindi provvede a rimuovere il client dalla coda e a processarne la richiesta.

## Protocollo di comunicazione client-server

Il protocollo di comunicazione tra client e server è stato progettato specificamente per fare in modo che il server non si blocchi in alcun caso ad attendere una risposta di un client. Per ottenere ciò, di ogni client connesso si mantiene uno stato, che serve al server per sapere che messaggio deve aspettarsi dal client. Ad esempio, in caso il client esegua un’operazione di scrittura, come prima cosa invia al server un pacchetto contenente il tipo di operazione che vuole eseguire (FCP\_WRITE), la quantità di bytes che invierà al server, ed il nome del file su cui scrivere. A questo punto, il server non aspetta attivamente che il client invii i dati, ma aggiorna lo stato del client e considera la richiesta soddisfatta. Il client poi invierà i dati con una seconda richiesta, ed il server saprà quanti bytes leggere, essendo questi stati salvati nello stato del client.