COMP 8005 Assignment 2 Design Document

Christopher Eng

Feburary 24, 2015

This document will explain the design decisions behind each module.

tcp_svr

Traditional servers are reliant upon multi-process and multi-threaded solutions in order to handle many requests. Since each recv request is blocked by a thread, I chose to create pools of threads, each controlled by a forked child process. The child processes were responsible for receiving new connections from the main process, as well as keeping track how many connections and threads existed. For performance reasons, a set number of processes and child threads were pre-allocated. More threads were added when the boundaries on how many connections the process could handle were being met. A small buffer of 5 unused threads was maintained, in order to allow connections to be handled quicker, rather than waiting for a new thread to be created. In the same way, a larger buffer of 10 unused threads was used to determine when to delete threads.

The main process, which accepted new clients, determined which child process should be assigned another connection. This is to balance the distribution of connections evenly among the processes. Processes were used for the main pools, to isolate the amount of memory used by each and to allow the server to be more failsafe, only crashing one process of many if there was a crash.

select_svr

The select server was designed similarly to the traditional server. However, due to time constraints, the design would have been changed to allow for fewer threads than clients. Since each thread is not blocked by a single connection, the distribution of handling requests would be allocated when the request came in. The select server was purely multi-threaded instead of using processes to manage the worker pools.

epoll_svr

The epoll server managed to fulfill the weaknesses in the select server's design. However, time constraints prevented this server from reaching completion.