Intro to Network Programming Discussions

The program can be broken down into 3 types of scenarios: multiple calls of write, writev, single call write that would be written into the buffers. Multiple writes would take the longest since the number of calls (15, 30, 60 for nbufs) would be equivalent to having multiple processes working on the same file descriptor. It would count the bytes, starting from the buf’s beginning to the current position of the file descriptor. The writev works on the same principles as write call, but it utilizes gather-scatter I/O system calls, which creates count segments from the buffers that are described by iov to file descriptor. Writev would utilize multiple buffers to be written in full. Finally, the single write call would be the fastest since the client would just need to call write once, which means only one segment, and the server would read that segment once.

If this program was placed in a scenario, where the network is 1 Mbps (slower than linux labs or local machines), then the times it takes to transmits and receive would take longer (longer delays) for all the writing functions. This can be to the limitations between nodes in the network for slower networks, which could mean that the nodes might be in poorer quality/condition and might cause data losses or corruptions since data is spending longer time in between nodes.

The reason why you would use threads over just servicing them in the main loop function is that the functionality that threads provide. Each thread has its own resources that can be used across the entire program like global data, code, and have their own stacks. In addition, it contains the program counter, execution state, that allows threads to know if the data is being sent into the stack appropriately.

