Question: What happens in your implementation if, during a PUT with a Content-Length, the connection was closed, ending the communication early? This extra concern was not present in your implementation of dog. Why not? Hint: this is an example of complexity being added by an extension of requirements (in this case, data transfer over a network).

Answer: In my implementation of PUT with a Content-Length, the server will read x amount of data (where x is represented by the content length) into the buffer then write the data to the file. Once all of the bytes specified by Content-Length is read and written to the new file, the fd for the new file will close and so will the connection between the client and server.

This is a result of modularity. As data transfer over a network is labeled as high complexity, modularity breaks up the process which will result in fewer bugs and interactions. This is seen with the case where the client closes the connection with the server once the server reads and writes up to the Content-Length. This in fact reduces the amount of interactions between the client and the server, and as a result, will reduce the amount of bugs compared to the situation where the client and server has an infinite amount of time for interacting with each other.

For example, if the client sends a request with a Content-Length to the server, but maintains the connection to the server then sends another request, the server may respond to the second request first as the first request may have timed out. This would in fact become an issue as the client wants the first request fulfilled before the second. As a solution to this complex issue, the client will close its connection with the server before sending in another request, which, maintains modularity.

The reason why dog.cpp does not concern this type of Implementation is because dog.cpp is one whole module. It does not rely on any other modules such as how the server relied on the client. Because of this, dog.cpp did not have the amount of complexity that httpserver.cpp did. As a result, dog.cpp did not need to terminate itself after it echoed whatever was in the Standard Input.