EECS 678: Lab 11 - The Unix Socket Protocol

Gehrig Keane
2727430

Sunday 1st May, 2016

Briefly describe the design of the program. How does your program control when the client runs and when the server runs?

The program starts the server first, followed by the client because the server needs to be prepared to hear the clients first communications. Once the client gets started, it establishes communication between itself and the server initiating the handshaking procedure. Handshaking arranges the communication channel in a reliable fashion.

What is the purpose of the handshake socket? Why not have the server create and bind session sockets that clients may connect to directly?

The handshake socket facilitates inbound connection management, ergo said socket establishes direct lines of communication between client signals. The handshake socket, as mentioned previously, provides an incoming connection interface for reliability of the system. Hard coding session sockets is fragile and doesn't scale in dense networking environments.

For the simple / client server program, we chose to use sockets instead of pipes to send messages between the client and server. Why are sockets preferred over pipes for this program? Give at least two reasons.

First, sockets are explicitly defined bi-directional communication and are suitable for singular, local, and dense network topologies. Additionally, adapting socket implementations between said paradigms requires minimal work. Pipes aren't designed to handle any amount of reliable networking.

Second, pipes aren't bi-directional by nature, therefore, they don't explicitly match the focus of the client-server dynamic. This would require a large amount of massaging pipes into use fullness, which in the system programming paradigm would probably result in a fragile/hacked-together implementation.