In this lab, you’ll practice your TCP programming skills by developing a simple proxy server.

Proxies

A *proxy server* is a special program that acts as an intermediary in web or other internet operations. For example, suppose I want to view a webpage on my computer (A). If B is the proxy and the webserver with the desired page is C, I aim my browser at the proxy on B. It forwards the request on to the webserver on C and then forwards the response to me on computer A. As far as my computer and the webserver are concerned, each is talking with B.

Why would I want to do this? We briefly talked about one application in 226: Network Address Translation (NAT) involves multiple computers sharing a single IP address in order to conserve addresses. The NAT server acts as an intermediary, forwarding the requests and responses for many computers which appear to be programs using different ports on the single server system. Other types of proxies can be used to centralize internet traffic for logging or caching common content. They can also be used to thwart centralized control, as when political dissidents use a proxy to reach websites blocked by their governments.

How does one make one of these amazing creations? The idea is simple. The server accepts connections. When one is made, it determines where that connection should forwarded to and initiates an outgoing connection with that destination. Once both halves are established, it merely forwards data received from each member of this pair of connections to the other member of that pair.

Write a simple proxy server. When your server starts, it should take two port numbers and an IP address. It listens on the first port number. When a client connects to it, the server opens a connection to the destination specified by the IP address and the other port number. Once the connections are established, the server assumes that data alternately moves between these endpoints. It receives from the client until it gets the byte 0, then receives from the destination until it gets a 0, then from the client, and so on. This behavior is custom made to serve as a proxy for the IM program from the last lab; verify that you can message through the proxy.

If you have additional time, you can improve the proxy. One idea is to have it take the destination from the first bytes of the message. Another is to use `select` in order to allow the proxy to receive bytes from either member of the pair of connections.