Network Programming
UDP Streaming Media Proxy

For this project, you will be constructing a streaming media proxy, similar to a SIP proxy used for VoIP. You will need to handle registration (keeping track of users' IP addresses), call setup (accepting calls and notifying proxy users when they have an incoming call), and media passthrough (passing media packets between users who are engaged in a call). **Use dual-stack AF_INET6 sockets for this assignment.**

Part 1: Registration

Registration and call setup packets will arrive on UDP port 34567. Registration packets will be in the format "REGISTER user_id". The user ID will contain alphanumeric characters only - that is, it will match the regular expression /^[A-Za-z0-9]+$/. When a register message is received, your server should record the peer address in a table, and respond with a message "ACK_REGISTER user_id".

Part 2: Call Setup

Calls will take place between two users of the proxy. When a user wishes to make a call, the user will send a message to port 34567 as follows: "CALL FROM:user_id TO:destination_userid". At this point, the call message should be forwarded as-is to the destination user if their address is known. If the address is not known, the calling user should receive a message "CALL_FAILED unknown peer". Otherwise, the proxy should expect a packet from the call recipient. This may be either "ACK_CALL FROM:destination_userid TO:userid" or "CALL_FAILED FROM:destination_userid TO:userid reason". Either of these messages should be passed through to the call originator. If the message is "ACK_CALL", proceed to set up the media path as described in part 3.

Part 3: Media Path Setup

Once a call has been acknowledged, the proxy needs to provide a path for the call's media (audio or video). This data will take the form of UDP packets. The proxy should create a new socket and use a new port number for this. bind() the socket to a port allocated from a predefined range (I'd suggest 5000-5999 - anything below 1024 will require root privilege). Then send "MEDIA_PORT FROM:user TO:other_user port_number" messages to each peer. Finally, start a thread. This thread should receive packets on the media socket and send them to the other user. The identity of the user who sent a packet can be determined by looking at the source address. In a real SIP proxy, we would need a mechanism to tear down this media path when the call completes, but for purposes of this assignment, you do not need to consider this.

Testing Hint

I will try to get a test client out ASAP, but in the meantime, on Linux you can use "nc -u" to send and receive UDP packets from the console. For example, run "nc -u localhost 34567" then enter packets one per line at the console to be sent to localhost port 34567. The replies will also be printed on the console. Note that packets sent this way will have a trailing newline. You can get around this by running something like "echo -n hello | nc -u localhost 34567", but this will limit you to one packet per nc invocation.