Programming Assignment 2 — TCP Packet Forwarding

Due: Friday, December 10, 2010

There are situations (mainly involving security restrictions) in which a process on some machine, A, wants to make a TCP connection to a machine B, but is not allowed to do so. For example, most web browsers only allow a Java applet included in a web page to open TCP connections to ports on the host from which the web page was loaded. If an applet found in a web page fetched from the HTTP server on some machine C tries to contact any machine other than C, the Java security mechanisms will reject the connection request.

In such a situation, there is a simple work-around. Assume that some process on machine A is not allowed to directly connect to machine B but can connect to a machine C which is allowed to make connections to B. One can run on C a process that forwards packets and connection requests between A and B. The idea is that the process running on C listens for connections on a pre-specified port. When it receives a connection request (from some A), it in turn makes a connection to B. After both connections are established the program on C merely forwards the data A sends on to B and forwards any data sent by B back to A.

For this assignment, I'd like you to construct such a program. Your program should expect three command-line arguments specifying

- The port number on which it should listen for incoming connections,
- The domain name of the machine to which it should forward data for any connections it does received (i.e. B), and
- The port number to use when connecting to the target machine.

You can test your program by running it with a target address and port number corresponding to a well-known port (HTTP=80, POP=110, etc.) on a local host. Then, use the Unix telnet program to connect to the port on which it is listening. If everything is done correctly, the responses you receive should be equivalent to those you would receive if you used telnet to connect directly to the chosen "well-known" service.

You may write the program in the language of your choice, but I would strongly suggest Java. The networking facilities are very simple to use. The chapter from Niemeyer and Peck's Java text that I made available earlier should provide all the information you need.

Your program should be designed to handle multiple simultaneous connections. That is if processes on two hosts A and A' contact your packet forwarding process it should establish independent connections to B for each incoming request and forward packets for both pairs of connections in parallel. It should also work regardless of whether the client or the server sends the first data. For example, with the (nearly defunct) FINGER protocol, the client sends data and then the server responds while with POP, the server welcomes the client (by sending a message) as soon as the connection is established and then the client responds. These features will require the use of Threads.