CSCI 315 Operating Systems Design
Fall 2014 - Prof. Felipe Perrone
Activity 25

Work on this activity with at least one partner in class.

The original source code that accompanies this activity can be found at **`~cs315/Labs/Net/Net1`** - what you have already done for the previous activity, though, will serve as a better starting point for today. Note that all the networking code developed for this activity should use UDP datagrams.

1) Copy **`send_udp.c`** to a new file called **`freq.c`** - based on this new file, you will create an a program that accepts a small set of command line arguments with behaviors described as follows.

    a) **`freq check "path/filename.ext"`**

    Sends a message to a server with a command to check whether the file exists in the given path or not. In case the file is found, **`freq`** prints to the terminal the message "**`File exists`**", Otherwise, it prints the message "**`File not found`**".

    b) **`freq get "path/filename.txt"`**

    Sends a message to a server with a command to request the transfer of the file. The server sends back the contents of the file using datagrams and **`freq`** saves it to the current working directory with the same name.

2) Copy **`recv_udp.c`** to a new file called **`fserv.c`** - this program will be a *server,* that is, it runs on an infinite loop receiving and handling requests for remote operations on files. The server receives datagrams that will contain well-formed requests. (You are in charge of determining the "shape" of these request messages.) If the request is **`check`**, the server needs to send back a trivial message. If the request is **`get`**, the server reads a "chunk" of the requested file (that chunk can be as big as you can fit into a UDP datagram), puts it inside the datagram and sends it back to the **`freq`**.

Obviously, you will need to make some design decisions in order to work on (1) and (2). Here are a few of them; you will need to reason through the other needs of this project. First, you need to create a simple application level protocol so that your commands can be encoded into a datagram for the **`fserv`** server. For instance, the first by in a request datagram could encode the command number; other parameters would come in subsequent bytes. Second, build your programs so that they use some random port number above 8,000 that is not likely to be used by your fellow classmates. Finally, think of how the server will communicate with **`freq`** when it sends back possibly multiple datagrams that together will build a file. UDP doesn't guarantee "in order" delivery of datagrams, so you will need to number the chunks that are sent back to **`freq`** and write them to file in ascending order. Also, you need to decide on how your server will communicate to **`freq`** the number of datagrams to expect to receive until the file transmission is complete. (For now, you may assume that there will be no loss of datagrams.)