Networking with Node

In this lab, you will learn to use Node’s low-level networking capabilities to create a simple TCP FTP (File Transfer Protocol) server and interact with it from a network client.

# Objectives

In this lab, you will learn to

* use various server & client functions of the net module,
* use the net.Server, net.Socket, and Buffer objects,
* write asynchronous callback functions to handle events from sockets, and
* create a simple FTP server.

# Write the server logic

We'll jump right into writing the server.

## Create a socket server

1. Create & store in a variable called server a new socket server via net's createServer factory function. For its callback, create an anonymous function with one parameter, the socket.

## Start listening on a port

2. Tell the server to start listening on port PORT, and provide an anonymous function to send a listen message to the console.

## Test the server

At this point, believe it or not, we have a functioning TCP server that can listen on port PORT, and accept connections & handle disconnections.

3. Execute server.js using node. You should see a message that looks something like the following.

$ node server.js

server listening on port 1986

4. Open another terminal window and start the client.js program with:

node client.js

This program accepts console input, sends the data to the server, accepts responses from the server, and writes that data to the console.

4a. Or open another terminal window and enter the command telnet localhost 1986. You should see output similar to the following.

$ telnet localhost 1986

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

Kill the server (via Ctrl-C if in a terminal) and you should see the client terminal echo something similar to the following.

Connection closed by foreign host.

When you are confident that the server is running and you're seeing these messages, move on to the next step.

## Handling Events

Use the ‘on’ method to define an event handler for the socket or server event emitter. For example, to register an event handler for the end event for the socket, use:

socket.on(‘end’, function() {

console.log(‘The socket is closing.’);

});

## Managing Server events

The server has one event, error, requiring you to register an event handler. Have the callback display the error object to the console.

## Managing Socket events

5. When the socket emits the data event, get the data from the callback, convert to a string (with the toString() method), trim() the whitespaces.

6. When the socket emits the end event, display a message to the console.

## Create the FTP Server

7. Evaluate the data when the user sends a command line to the server. The commands should be quit, ls, dir, and pwd. Use the fs object to access the file system. Use fs.readdir(dirname, function(err, fileArray){}) to get the contents of the current folder. Use fs.realpath(dirname, function(err, path){}) to get the complete folder name.

8. Congratulations, you have finished the lab.

## Extra Credit

For extra credit and fun and relaxation, add the following commands to our simple FTP server. Note that some commands require that you also create an FTP client to accept the data from the server and do something with it.

Additional commands to add:

* GET filename - get the file contents from the server and send back to the client
* PUT filename - get the file contents from the client and save on the server
* CD foldername – change the current folder from ‘.’ to another folder. NOTE: this will require that each connection maintain a separate current folder.