OREGON STATE UNIVERSITY

ECE 406 PROJECT PAPER

# Logging Server

*Author:*
Andrew Collins

*Instructors:*
Matt Shuman

June 11, 2019

**Abstract**

This paper describes a generalized model and implementation of a logging server using socket programming. It covers the setup required for implementing a simple server that can communicate with multiple clients and a version that can handle multiple clients in parallel. Finally, an additional feature that allowed data to get pushed to the cloud was added.

## I. Introduction

THIS document has been written as a tutorial, not a complete reference. We shall first go over all the hardware we will be using in the project, then introduce this projects objectives, and implementation of each objective.

A server provides an array of functionality, but for this project the main tasks of the server will be to collect and log data in a efficient manner. This data can either be transmitted on a wired or wireless interface allowing for flexibility in the available communication infrastructure.

For example, let's say you wanted to measure, every hour, the temperature in every room of your house for a week. With a logging server you place a wireless temperature sensor in each room and have them send data to a server that will log all that data and have it instantly available.

## II. Hardware

This project requires you have a wireless network setup via a router that has access to the Internet. The router must be set up to support WPA/WPA2 wifi security.

The server code for the project was written in Python3 and was run on a macOS PC. The clients for this project consisted of two ESP8266 NodeMCU 2nd generation / v1.0 / V2 development boards and two Adafruit Feather M0 WiFi development boards.

To be able to implement cloud storage, we set up a Google account with activation of certain application which will be discussed in the further sections.

## III. Objectives I

### *A. Overview*

Objective 1 is the foundation of the whole project, it states that our system be able to handle communication between multiple clients and the server. So step one is to create a server and test it using a few client modules. Figure 1 below shows the parts of our model, which is know as a client-server model. This model has two nodes that communicate with each other to exchange some information. One of the two nodes acts as a client process, and another process acts as a server. For objective one our server

must be able to communicate with multiple clients and so we will have four clients that will talk to a single server.

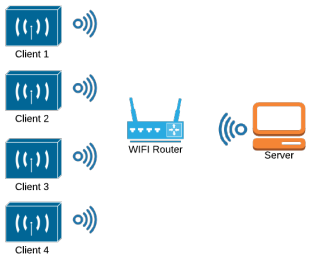


Figure 1: Client-Server System Model.

### *B. Implementation*

In order to create our client-server model we will use socket programming. Sockets allow communication between two different processes or applications on the same or different machines using standard Unix file descriptors[1].

The type of socket we will use is know as ”Stream Sockets” or ”SOCK_STREAM”, stream sockets are reliable two-way connected communication streams. Stream Sockets provide reliable, error free communication. Packets arrive in the same order they are sent.

Stream Sockets use a protocol called ”Transmission Control Protocol”, otherwise known as ”TCP”, this protocol makes sure your data arrives sequentially and error-free. ”TCP” together with ”IP” forms the ”TCP/IP” communication protocol where ”IP” stands for ”Internet Protocol”. IP deals primarily with Internet routing and is not generally responsible for data integrity[1].

Both the server and client create a socket with a unique IP and port[2]. The server creates its socket and listens for connection to its socket. The client must know the information of the servers socket, it then initiates the communication with the server using the servers socket information. The server accepts the connection request and now the connection between the two is established. Data can now be passed between them. The connection is closed by either parties. Figure 2 below shows the programming interface flow required to set up a client server connection oriented application. When we have multiple clients trying to connect to the server the additional clients will be placed in a queue until the server has finished processing the current client. The number of clients that can be in the queue is set by passing the desired value to the listen() function below.

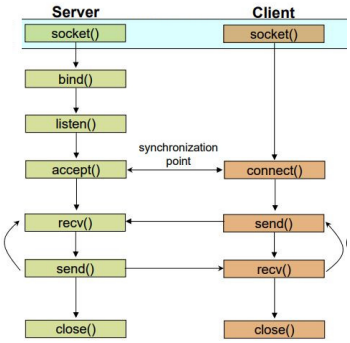


Figure 2: Application programming interface for a stream/connection oriented client server application.

## IV. Objectives II

### A. Overview

Objective two of our project is to add additional features to our server. Specifically we are going to give our server the capability to process multiple clients in parallel. The server from objective one is know as a ”Iterative Server” it only handles only one client at a time, additional client must wait for all previous requests to complete[3].

A ”Concurrent Server” allows clients to connect anytime to the server, which uses multiple threads or processes to service connections. This allows the server to maximize response time for each client and increases throughput or number of clients processed.

### B. Implementation

To get the concurrency feature on our server we will utilize threads. Threading is a feature usually provided by the operating system. Threads allow multiple concurrent execution contexts within a single process. We implement our server as a single process with multiple threads, where a thread is created for each client that connects to the server. The _thread module and threading module is used for multi-threading in python. The function thread.start_new_thread() is used to start a new thread and return its identifier. The first argument is the function to call and its second argument is a tuple containing the positional list of arguments[4].

## V. Objectives III

### A. Overview

Objective three gives our server cloud storage capabilities. All the client data passed to the server can now be push to a Google Sheets file on the web. This allows us to monitor or access our data from anywhere.

### B. Implementation

In order for us to be able to push data to Google sheets we are going to need gspread on our server which is a Python API that allows you to manage your Google Sheets. On our PC we ran the command ”pip install gspread” to install the API.

Also in order for us to use the Google APIs we will need to use the OAuth 2.0 protocol for authentication and authorization. We can obtain the OAuth 2.0 client credentials from the Google API Console. The steps to getting credentials and using the protocol is a little complected and is best understood by reading and following the ”Using OAuth 2.0 to Access Google APIs”[5] or referring to the article ”Google Spreadsheets and Python”[6] by Greg Baugues.

## VI. Testing Procedure

In the server file it might be necessary to change the host IP to match the PCs IP. For the server we will use port 8090.

In order for the server and clients to communicate they must all be connect to the same Local Area Network. Inside the client sketch files it may be necessary to change the network SSID and password so the client can connect to the network. Also inside the clients sketch files we will need to match the same host IP address, ”HOST_IP_ADDR”, and port number we used for the server above.

***Setup Part 1:*** *Server:*

1) Open a command line interface, such as Terminal
2) Navigate to the Server directory.
3) Run command ”pyhton3 server_mult_thread_push.py”.

***Setup Part 2:*** *Client:*

1) Open each client sketch code in the Arduino IDE.
2) Enter the networks SSID and password in the indicated sections in the code.
3) Enter the servers IP address and port number obtained from the previous section, in the indicated sections in the code.
4) Inside the Arduino IDE upload the sketch file to the development board.

## VII. Acknowledgment

The author would like to thank instructors Matt Shuman and Benjamin Brewster for all the help and advice offered towards the course.

## References

[1] B. Hall, “Beej’s guide to network programming: Using internet sockets,” Jun 2016. [Online]. Available: https://beej.us/guide/bgnet/html/multi/index.html

[2] P. Fatourou, “Introduction to sockets programming in c using tcp/ip,” May 2012.
[3] B. Brewster, “Ece 344 operating systems 1: Network servers.”
[4] S. UPPAL, “Socket programming with multi-threading in python,” Sep 2017. [Online]. Available: https://www.geeksforgeeks.org/socket-programming-multi-threading-python/
[5] “Using oauth 2.0 to access google apis — google identity platform — google developers.” [Online]. Available: https://developers.google.com/identity/protocols/OAuth2
[6] G. Baugues, “Google spreadsheets and python,” Aug 2018. [Online]. Available: https://www.twilio.com/blog/2017/02/an-easy-way-to-read-and-write-to-a-google-spreadsheet-in-python.html