Cal State Fullerton

Computer Science

### **CPSC 471: Network Programming**

### **Project Report**

### **Spring 2024**

#### 

#### 

#### 

#### 

#### **Submitted by:**

Dominic Jacaboo

Jair Chavez

Shahmir Ahmed

Ahmir Bijanzadeh

Miles Chao

**Submitted to:**

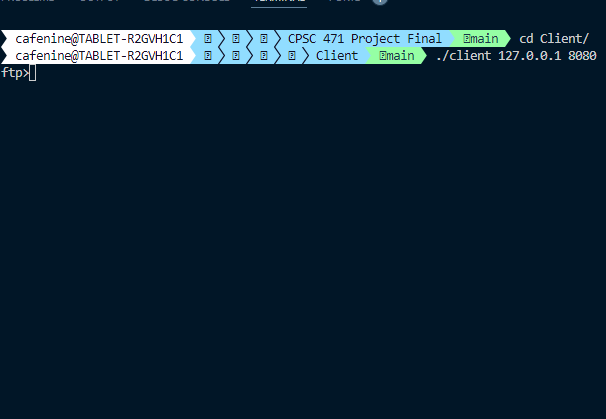
#### Professor Kurban

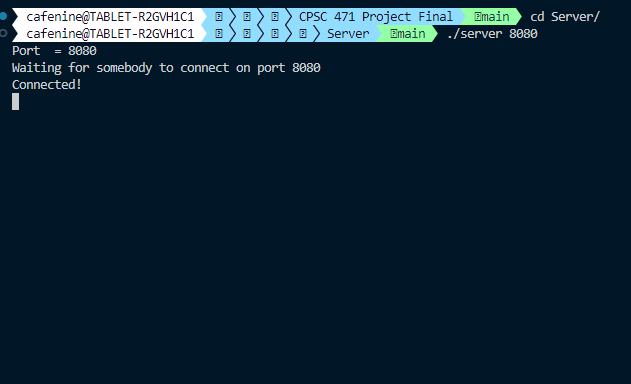
# Table Of Contents

1. **Project Demo**
   1. Demo Connection to Server
   2. Demo of Implemented Functions
      1. Put
      2. Get
      3. Ls
      4. Quit
2. **Protocol Design: Application-layer Protocol C++ TCP Communication**
   1. Message Types Transmitted
   2. Responses from the Server
   3. Sizes and Format of Messages
   4. Settings up Sessions and File Transfers
   5. Handling the Transmission of Files
   6. Managing TCP Buffers
3. **TCP Protocol Diagram**
   1. Client-Server Interaction
4. **Libraries / Modules**

# DEMO Pictures

## Demo of Initial Connection:

****

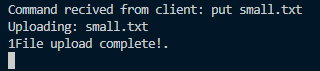
****

## Demo of Put:

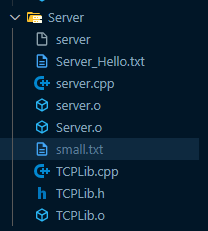
Client



Server



Output

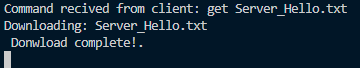


## Demo of Get:

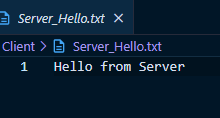
Client

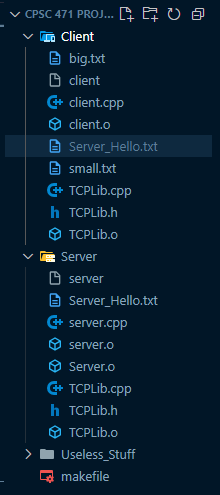


Server



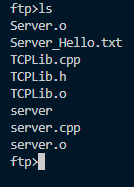
Output





## Demo of ls:

Client

****

Server



## Demo of Quit:

Client



Server

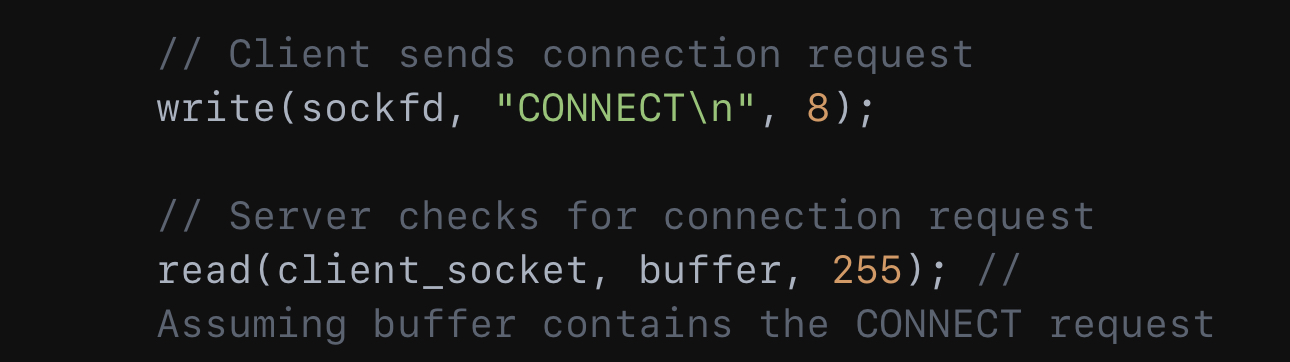


# 

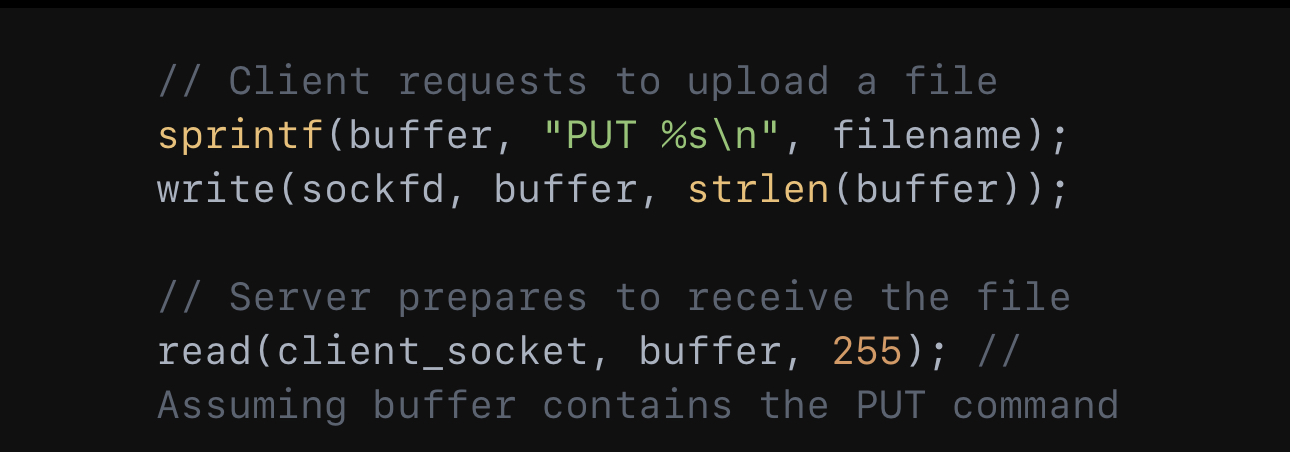
# Protocol Design: Application-layer Protocol C++ TCP Communication

## 1. **Message Types Transmitted**

* Connection Requests and Acknowledgments: Upon successful connection, clients send a "CONNECT" request to the server, which starts the communication process.



* File transfers can be started with the help of commands such as PUT, which uploads files, and GET, which downloads them.



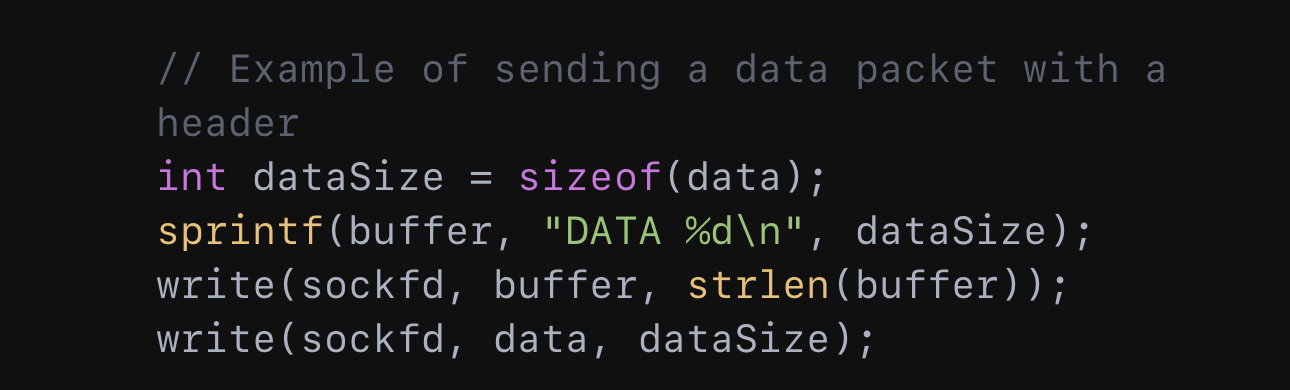
## 2. **Responses from the server**

* In order to control the flow and guarantee data integrity, servers reply to connection requests, file transfer commands, and data packets with the proper acknowledgments.



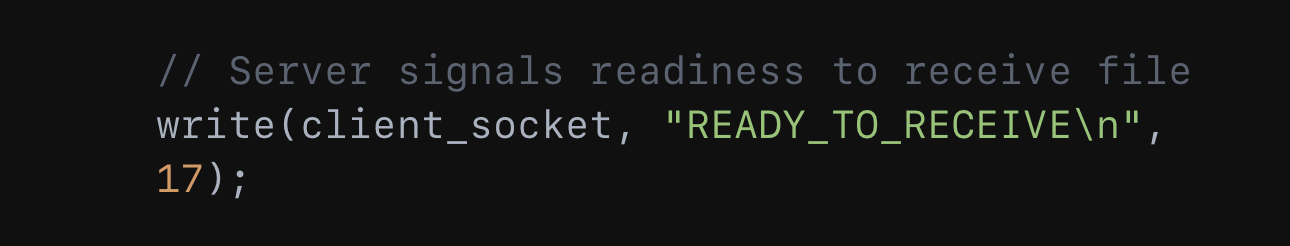
## 3. **Sizes and Formats of Messages**

* Both basic command strings and more intricate data packets are transmitted in the messages:
  + Control messages: Short strings that contain parameters and commands, finished with a newline.
* Data packets: Binary data is sent first, with headers indicating the content length.



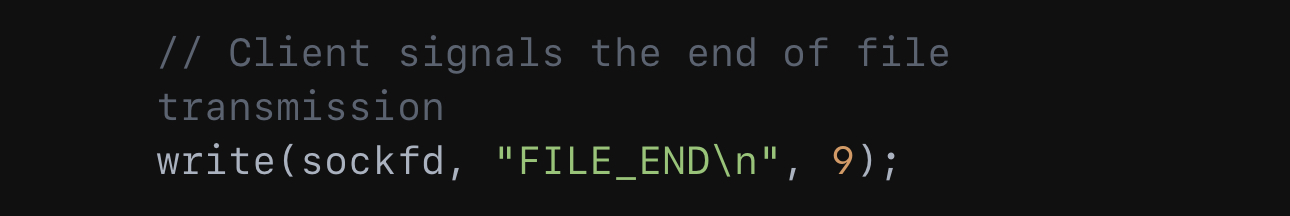
## 4. **Setting Up Sessions and File Transfers**

* The PUT or GET command initiates a file transfer session, which is facilitated by server readiness signals such as READY\_TO\_RECEIVE or READY\_TO\_SEND.



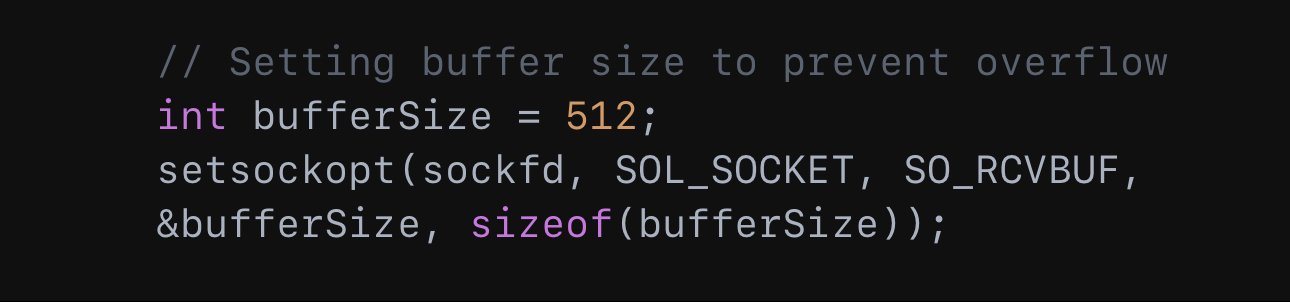
## 5. **Handling the Transmission of Files**

* Start and Stop Signals:
  + When the server delivers a FILE\_START message, transmission begins. It ends when it receives a FILE\_END signal, indicating that all data has been sent.

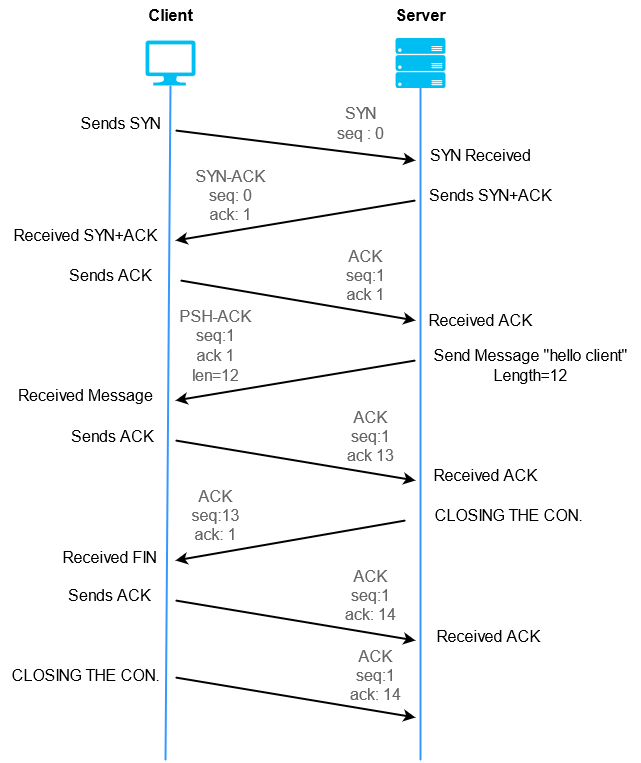


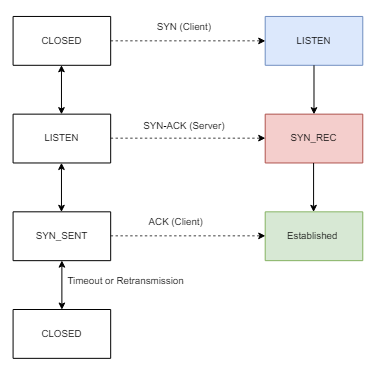
## 6. **Management of TCP Buffers**

* The system makes use of flow control techniques including waiting for acknowledgments (ACK) before sending further data, as well as customizable buffer sizes, to prevent buffer overflow.



# TCP Protocol Diagram C++



****

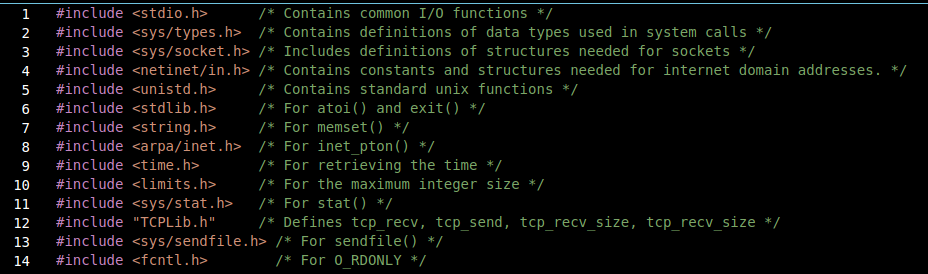
# Modules for Application Layer Protocol and Protocol Diagram Creation

This chapter's modules on designing application-layer protocols and making protocol diagrams are from Chapter 3: Transport Layer. Understanding the workings of upper-layer protocols such as FTP requires an understanding of Reliable Data Transfer (RDT) protocols, which describe how data is consistently transported over a network. The use of finite state machines (FSMs) to visualize and design protocol behavior is demonstrated. We used/referred to these specific slides: Slides 66–69: These use state diagrams and FSMs, which are essential for protocol design, to explain the idea of dependable data transmission protocols. These slides aid in the comprehension of the protocol's state changes and operation sequence. Slides 70–71: These include in-depth descriptions of the Go-Back-N and Selective Repeat procedures, among other pipelined protocols, and how they work. These slides are very helpful in demonstrating the difficulties in creating protocols that guarantee dependable and effective data transport. These slides helped us create the application-layer protocol for our project.

The particular slides we used from the modules for the project are from "Chapter 6: Link Layer," and they are as follows: Ethernet's functions, such as how devices control access to the network medium, on slides 6-56 to 6-58. Slides 6-63 through 6-68: Explain how switches learn and forward data throughout a network, with an emphasis on switching technology and MAC addresses.

# **Libraries/Modules Utilized:**

C++ Libraries used in our code and a brief explanation of their purpose or where in the code

they are being referenced. These can be found in the client.cpp file, while the server file includes all of the same up to <limits.h> including TCPlib.h.

Other modules:  
A makefile that contains some commands for opening the socket and creating or destroying the server/client files.