

Yen-Ju Tseng

tyj850916@gmail.com | +1(858) 729-3110 | www.linkedin.com/in/yenjutseng | <https://tyj99.github.io/>

EDUCATION

University of California San Diego, Jacobs School of Engineering

San Diego, USA

Master of Science in Electrical and Computer Engineering

Sep 2021 – Jun 2023

- Coursework: Software Foundations, Operating Systems, Computer Networks, Web Client Languages, Graduate Networked System, Advanced Data Structure, Principles of Programming Languages

National Taipei University

New Taipei City, Taiwan

Bachelor of Science in Communication Engineering

Sep 2015 – Jun 2019

- Coursework: Data Structure, Advanced Computer Programming, Database System

SKILLS

Programming: C++, Golang, Java, Python, Kotlin, JavaScript, HTML/CSS, MATLAB

Tools: Git, Visual Studio, Visual Studio Code

PERSONAL PROJECTS (code available upon request)

Relational Database System in C++17

Apr 2022 – June 2022

- Design and build a working relational database like MySQL using C++17.
- This system involved data interpretation, data manipulation, data querying, and showing table data results. Various software design patterns were used during development.
- The database system is built upon the **MVC(Model-View-Controller)** application design pattern.
- Use Scanning, tokenizing, and parsing to handle user input.
- Use the **chain-of-responsibility** design pattern to handle processing of user provided commands.
- Use the **factory** design pattern to handle statements.
- Use indexes and **LRU (Least recently used cache) Cache** to improve this database system.

Sliding Window Protocol in C

Jan 2023 – Feb 2023

- Implementing communication between two or more hosts with **sliding window protocol** that uses **selective repeat/retransmission** and **cumulative ACK** to ensure the **reliable in-order** delivery of frames between hosts since frames sent on the network links can and will be corrupted in flight. (window size = 8 on both ends)
- Sender hosts must transmit messages typed in at the command line to a corresponding receiver host.
- Receivers need to reassemble frames, retrieve the correct message that the sender had sent, and output it.
- Divide the messages which are larger than `MAX_FRAME_SIZE` (i.e. 64 bytes) into frames.
- Implement **Error Detection Mechanism** which is **CRC-8** on senders and receivers.
- Establish Connection before Sending Actual Data: SYN, SYN-ACK.
- Add **sequence number**(uint8_t) and the implementation continues to function correctly during the **wrap-around scenario**.
- A sender may communicate with only one receiver at a time, but a receiver must be able to handle frames from multiple senders at the same time.

Building a Simple Router in C

Feb 2023 – Mar 2023

- Building a simple router. It will receive raw Ethernet frames and we handle those packets with **ARP request, ARP reply, ARP caching, ICMP** (send messages back to a sending host), **Switching, Longest Prefix Match, IP sanity-check** (minimum length and checksum), and the other essential features for IP forwarding.
- Support **ping** and **traceroute** for both clients and servers, and download a file using HTTP from one of the app servers.
- Implement **Longest Prefix Match** using **Trie**.

Fault-tolerance Scalable Cloud-Based File Storage service in Golang

Jan 2023 – Mar 2023

- It is a networked file storage application that is based on **Dropbox**. Multiple clients can **concurrently** connect to the server to access a common, shared set of files. A client can interact with the service via **gRPC**.
- A file in the server is broken into an ordered sequence of one or more blocks. Use the **SHA-256 hash function** for each block, and the set of hash values represents the file, and is referred to as the hash list.
- Use **protocol buffer** for **gRPC**, and **versioning** and hash list for handling **update conflicts**(with some logics)
- A client program will create and maintain an **index.db** file in the base directory to help **sync** operation.
- Implement a mapping approach based on **consistent hashing** for block storage to make the server **scalable**.
- Make the server **fault tolerant** based on the **RAFT distributed consensus protocol** (implement the **log replication** part of the protocol)

Building a Simple Web Server in Golang

Jan 2023 – Feb 2023

- Implement a subset of the **HTTP/1.1 protocol** called TritonHTTP. It is a client/server protocol that is layered on top of the reliable stream-oriented transport protocol **TCP**. (Only implement the **GET method**)
- Implement **HTTP pipelining, HTTP persistent connection**, and **virtual hosting** by allowing TritonHTTP to host multiple servers.