Yen-Ju Tseng

tyj850916@gmail.com | +1(858) 729-3110 | LinkedIn Profile| Personal Website

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/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.

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** (**log replication** part of the protocol)

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. (window size =8 on both ends)
- Divide the messages which are larger than MAX_FRAME_SIZE (i.e. 64 bytes) into frames.
- Receivers need to reassemble frames, retrieve the correct message that the sender had sent, and output it.
- Implement **Error Detection Mechanism** which is **CRC-8** on senders and receivers. A sender may communicate with only one receiver at a time, but a receiver can 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**.

Nachos Operating System Implementation in Java Sep 2022 – Nov 2022
- Implement **Alarm class**, including **waitUntil(x)**, **timerInterrupt()**, and **cancel()**. Also, implement **KThread.join()**.
- Implement **condition variables** using **interrupt disable and restore** to provide atomicity, including **sleep(), wake(), wakeAll(), and sleepFor(long x).**
- Implement the **Rendezvous** class to provide a mechanism for threads to exchange values, using **locks** and **condition variables** to manage **concurrency**.
- Implement the file system calls which are **create**, **open**, **read**, **write**, **close**, **unlink**, **exec**, **join**, **exit**, and **halt**.
- Implement support for **multiprogramming** by managing the allocation of pages of physical memory so that different processes do not overlap in their memory usage.
- Implement **Demand Paging, Lazy Loading, and Page Pinning.**