# CS 4713
# INTRODUCTION TO THE INTERNET: ARCHITECTURE AND PROTOCOLS

## ASSIGNMENT: 1

I tried using a top-down approach for this assignment, possibly defining my functions before implementing them. This approach saved me a lot of time and helped my code to remain organized as well. I defined separate header files so that anyone can make sense of my code.

Whenever we receive a packet, it's either an ARP packet or an IP packet. This differentiability makes it easier for us to handle incoming packets.

- If it's an IP packet, I performed sanity check and verified its checksum. After this, I checked if it's for one of our interfaces.
    - If it's for one of our interfaces, we drop the packet (for a TCP or UDP packet). If it's an ICMP request, send its reply.
    - If we need to forward the packet, find its outgoing interface and look for next hop MAC address in the ARP cache. If it's not there, generate an ARP request and buffer the packet in ARP queue, otherwise send the packet using the IP->ARP mapping that we found in ARP cache.
- If it's an ARP packet, perform the sanity check and figure out if it's an ARP request or reply.
    - For the case of an ARP request, cache it and send its reply if it's for one of our interfaces.
    - If we received an ARP reply, cache the mapping and send all the packets that are waiting on this reply.

In this assignment, I tried modifying the packets that are already in flight whenever I needed to send a reply etc. but I faced difficulty handling a lot of pointers. To solve this problem, I made new headers whenever I needed to forward packets and changed their fields. Although, this implementation may not be the fastest, but it helped me avoiding any pointer errors in my code.

Comments have been added for each function that I've created or modified for this assignment.

The assignment implements the functionality of sr_solution as described in the manual and all test cases are working.