# COMP9331 Assignment 1

## 1 Overview

This assignment is implemented with python 3. To achieve the goal, implementing a P2P network based on circular DHT, of this assignment, I imported threading and socket modulars into my work. Threads for UDP server, TCP server, input listenning and timers for successor checking are running simultaneously. Additionaly, there might be sub-threads running to allow multiple clients connecting to a TCP server. I also created a few files:

1. P2P.py : containing all methods of this implementation.
2. init_python.sh : establishing an initial P2P network.
3. join_python.sh : creating a new peer and joining into the pre-existed network.
4. 4103.pdf : can be requested and sent.
5. README.md : A simple tutorial of how to run the code.

## 2 Implementation Methods

1. A class of P2P is defined, which contains almost all functions.
2. The first thread is for UDP server, which loops forever to handle all the UDP messages and to modify the recording of successors and predecessors of a peer.
3. The second thread is actually a timer. For every PING_INTERVAL, the udp_client sends a ping request to its successors and counts consequtively dropped messages.

4. The third thread is also a timer, which is to check if a successor is still alive. The MAX_LOST_PACKET is set to 2, which means once a successor is not answering for 3 times, it will be regarded as a dead peer. Also, the successors will be updated.

5. The forth thread may even split into more threads, because it is designed to serve multiple clients. The sub threads handle all the TCP messages. They send necessary TCP messages too.

6. The fifth thread is a self loop, which gets input commands from users.

7. The tcp_send* functions are used as temporary clients, which will be aborted once the message/file is sent.

8. The main entrance of this program takes arguments provided by scripts to initialize an object of P2P class, which records necessory information and run functions mentioned above.

# 3 Advantages

1. By implementing sub-threads, the server can actually serve multiple clients without any confliction.

2. By implementing timers, the frequency of sending UDP messages does not rely on the running time of any function. Comparing to method of time.sleep(PING_INTERVAL), this timer method is much more accurate, which means a departure of a peer can be detected a little bit sooner.

3. When sending or receiving a file, we read or write it as a binary file, which allows us to send/receive files with any kind of extensions, including .pdf, .html, .jpg, .exe etc.

# 4 Disadvantages

1. Using while loop to keep a thread running can cause problems when you need to exit properly. The loop usually stucks because it does not check the looping condition when running 'the' loop. (Though it does not infect anyting in this program, it is still pretty annoying.)

2. I used temporary client in this program. In other words, I did not send messages and wait for the reply of server, which is not efficient enough. This methods wastes time bcause s.accept() occurs many times.