Hao An 5138496

COMP9331 Assignment Report

Python version: 3.6

YouTube link: https://youtu.be/r8OSq7DCSRI

1 Description

There are five threading for Ping first successor, Ping second successor, UDP server, TCP server and monitoring standard input.

1.1 Ping successor

1.1.1 Two clients for sending Ping:

Send a message to its successor by using UDP **every 10 seconds.**
Message design:
'P ID 1'. Here ID is sender's ID and 1 means this message is send to first successor.
'P ID 2'. Here ID is sender's ID and 2 means this message is send to second successor.

1.1.2 UDP server:

A UDP server waits for receiving UDP packet. When server receives a Ping request, reply a response message. When server receives a Ping response message, just display on the terminal.
Message design: 'ID'. Here ID is receiver's ID.

1.2 Requesting a file

1.2.1 Monitor standard input:

If input is "request X", this peer will send a request file message to its first successor by using TCP. Here is a function for get the file name's hash value. We will send hash value to its first successor.
Message design: 'R file filename ID'. Here R means request, file is hash value, filename is requesting file name, ID is requester's ID.

1.2.2 TCP server action for this part:

If TCP server receive a message for requesting a file('R...'), it will check whether this file is here. If file is here, send a message to requester by using TCP.
Message design: 'F ID filename'. Here F means find file, ID is file owner's ID, filename is file name.
If file is not here, TCP server will send the receiving message to its first successor without modification by TCP.
If TCP server receive a message('F....'), it means find the file location and

display it on the terminal.

1.3 Peer departure

1.3.1 Monitor standard input:

If input is “quit”, this peer will depart from the network in a graceful manner. It will send message to its two predecessors by using TCP.
Message design: ‘D ID first_successor second_successor’. Here D means departure, ID is departing peer’s ID and successors are departing peer’s successor.

1.3.2 TCP server action for this part:

If TCP server receive a message(‘D.....’), if departing peer is its first successor, update its two successors. If departing peer is its second successor, update its second successor.

1.4 Kill a peer

1.4.1 Detection mechanism:

Here I set a counter. I just monitor peer’s first successor. If peer send a Ping to its first successor, the counter add 1 and if peer get a Ping response message, the counter will minus 1. That means if peer’s first successor is alive, the counter will be 0 after receive Ping response.

If counter > 3, I will say that peer’s first successor is no longer alive. Then send a message to its second successor to query second successor’s first successor by TCP. Then peer can update its first successor and second successor.
Message design: ‘N’. N means peer will get new successor.

After that, the peer will send a message to its first predecessor by TCP to tell that your second successor left.
Message design: ‘U first_successor’. U means you need update your second successor. First_successor is peer’s first_successor.

1.4.2 TCP server action for this part:

If TCP server receive a message(‘N’), it is to query my second successor, so send my successor’s ID as a reply.
If TCP server receive a message(‘U....’), that means this peer need update its second successor, so update its second successor.

2 Improvement

This assignment does not involve in Internet, just send TCP and UDP from local host to local host, so all message actually cannot lose. In my program, I don’t set timeout. if TCP response lose, the TCP connection will wait for

response all the time.

In this program, I just monitor each peer's first successor whether is alive. If its first successor is not alive, it will query its second successor. If its first successor and second successor are no long alive together, this program does not work.

Also, I use five threading, when I test my code, if I request file very frequently and run them longer, I find the input command will not work. When you type 'request' or 'quit', the terminal do not have reaction and no print there. I think it may be the multithreading concurrency problem.

In my program, we just consider peer departure, but in real p2p network, a peer joining into network is as important as a peer departure. We can build this part to improve this.