COMP3331 Assignment Report

Yu Xia z5212108

May 22, 2018

Python Version: Python 3.6.4

Screencast Link:

`https://www.youtube.com/watch?v=mO_RlEchgQY&t=195s`

Please click the *cc* button to open the subtitle, because I provided some comments and information in subtitle

The steps I take to implement the pragram:

First, each peer need to send ping request, while at the same time receive other peers' request and send corresponding response. That means each peer is both a client and a server. So I choose to use multithreading in python, where one thread is client and another one is server.

But soon I realise that only two threads are not enought. Because the assignment requires us to send ping request and response messages using UDP socket, while other messages are using TCP socket to transfer. I searched that UDP server and TCP server can listening to the same port, so naturally, I came up with the idea that I should use four threads running in parallel: UDPclient, UDPserver, TCPclient, TCPserver. Also, I have another thread that always be ready to take user input.

This is how my program works: first, send ping messages to two successors and wait for response, at the same time, keep track of the number of lost packets, if the number of lost packets is greater than or equal to 3, that means the peer has left. Then its two precedessor will print appropriate message to command wiindow and update new successor.

If the ping message went successfully, no peer is leaving the network, and it comes to requesting file part, my program will forward the request message to next peer until the file is found. When a peer is quitting, it will use TCP socket

to communicate its two predecessors, indicate that this peer is leaving, and also attach its two successors along with this message. That's basically how my program works and the steps I take to implement them. Next, I will talk about the design chooses I make.

Desing Decision

I choose to send ping message every 20 seconds, because I found if send ping request very often, it is hard for users to read the actual ping message. And more importantly, the server will deal with the ping request all the time, it will occupy too much resources and will slow down the whole program. On the other hand, if we send ping request messages not frequently, then it would take very long to find out whether the peer is leaving. So I think send ping messages every 20 seconds would be a good choose.

In my program, the number of lost packets before assuming peer is leaving is 3. On the one hand, we all know that UDP is not stable and it would sometime lost the packet, but the chance for that is very small. On the other hand, if we assume peer is leaving only because 1 or 2 lost packets, the decision is too easy to make, we cannot guarantee the peer is leaving. So, I choose 3 lost packets.

Message Format

Ping request message format:
message = "A ping request message was received from peer ..."

Ping response message format:
response = "A ping response message was received from Peer ..."

File request message format:
message = "File request message for ... has been sent to my successor."

response = "File ... is not stored here. File request message has been forwarded to my successor."

receive = "Received a response message from peer ... , which has the file ..."

Peer quit message format:
message = "Peer ... will depart from the network."