COMP 9331 Computer Networks and Applications Assignment

Yu Han Z5219071 25th April 2019

Outline: Class CDHT includes all required functions for this assignment. There are 9 parts of this class, which are init, Ping, TCP, sendPingMessage, sendTCPMessage, sendFile, receiveFile, and user_input. The python version is 3.7.

Init: The peer denotes the current peer, succ1 for successor 1 of current peer, succ2 for successor 2, pred1 for predecessor 1, pred2 for predecessor 2, MSS for maximum segment size, dropRate for drop rate of transferring files, fileLength used for record the size of transfer file received, fileDest used for record the destination peer of transferring files, run is the trigger of Ping thread and TCP thread, start_time used to record the start time. There are four threads in this application, which are threadPing for receiving UDP message, threadTCP for receiving TCP message, threadFile for receiving files (with UDP protocol), and threadInput for receiving user input.

Ping: This part is used for receiving response and request of peers. It is using UDP protocol. It sends request to successor in every 10 seconds, and also sends sequence number to its successors. succAck means how many times does the successor received the request, if the successor doesn't response more than 4 times, current peer will make this successor as leaved successor and will send "search" request with TCP protocol to its second successor to update its successor. This function will receive two types of message, it will send response message if it received "request" message, else just print received response if it received "response".

sendPingMessage: This part is used for encoding message and send it to destination peers with UDP protocol. The message includes the sender port, sequence number and request content.

TCP: This part is used for receiving peer quit request, update successor request, and file request of peers. It is using TCP protocol. There are two main purpose for this function. One is update peer's connection. If it received 'ask' message, it will update new successors for current peer if it's old successor going to 'quit'. Hence, the 'search' request will help current peer to find new successor, and 'response' message will notice current peer what is the number of its new successor. Another type is file request. If the message is 'file', means

some peer is requesting for a file. It will use file hash method to search the location of this file and send the file back if this peer has the file or forward the request to its successor if the file not in here. If received 'found', means the file has been found and will be transferred to this peer soon.

sendTCPMessage: Similar like 'sendPingMessage', it also used for encoding message and send it to destination peer but with TCP protocol. There are three types messages, 'ask' for search or update successors, 'file' for requesting a file, and 'found' for notice original peer that the file has been found.

fileLocation: This part is used for determining the file's location based on file hash number. If the file hash number is greater than 255 means it can be divided one more time, so return false. Else if the file hash is greater than current peer and successor is less than current peer, means current peer is the largest number in this circular, therefore, return true. Or if the differences between file hash and current peer is less than the differences between file hash and successor or predecessor, means current peer is the closed peer of the file then return True.

receiveFile & sendFile: This part is used for file transfer with UDP protocol and stop-and-wait method. Once the peer is requested to send the file, it will send the file size at first to inform the original peer the stop point. And then the original peer will request the file content and file peer will start to send the file with sequence and acknowledge numbers. If the file peer received 'next' message, it will send next part of the file, and it will stop send if it received 'finished' message. The file sender peer will generate a responding log of the sending record, which includes the sending file to original peer info, receiving response from original peer info, drop event happened during the transmission, and the retransmit event after drop event. The file receiver peer will generate a receiving log which includes the receiving file recording and send response message recording during the file transmit period. Normally, the receiver will divide the content received from the file sender to two parts, one is header which includes the sequence number and acknowledge number, another is the file content. The receiver will send message 'complete' to inform the sender that the file has been received.

User_input: This part is used for getting request from users. It can get the request file command or quit peer command while user input 'request X' or 'quit'. And it will forward

the file request to other peers to send the file. Or inform other peers to update successors because the current peer will leave. Also, it will change the self.run to False to stop the Ping thread and TCP thread once it received 'quit' request.

Design Trade-offs: I used multiple threads in this program for receiving UDP message, TCP message, transfer file, and user input. Transfer file part also used UDP protocol, so I used different port to contact to avoid conflict with Ping part. Also used stop and wait method for reliable UDP transfer, and compare sequence and acknowledge number for to avoid drop any data or message.

Improvement and extensions: Although it called peer to peer, it doesn't do real peer to peer things in this program. Only transferred the file from one peer to another. Share the file to other peers and let the original peer receive it from others will be more efficiency in real world for data transmit. Also, I think the number of bytes of those two log.txt files should not always be the MSS. If the rest data is less than MSS, the number of bytes should be the real size. However, I just followed the examples, set it as MSS.

Reference part: For the Ping and TCP part, I read many articles and examples which are related the UDP and TCP protocols. Also use Lab2 and Lab3 as examples to writes those two parts.

Video Link:

https://youtu.be/EMA0V0a3C0g