Ruijun Zhou z5223541

CDHT Assignment Report

Environment:
Developed using Python 3.7.0 on macOS
Test using Python3.7.0 on lab machine

Steps:

1. Wait for peers to ping each other and receive request pings from predecessors.
2. Request 2012 on arbitrary one peer except the one already having file 2012.
3. Choose one peer to quit politely.
4. Kill one peer and wait for its predecessors to find out that this peer has been dead.

Setup script:

```
xterm -hold -title "Peer 1" -e "python3 cdht.py 1 3 4 300 0.3" &
xterm -hold -title "Peer 3" -e "python3 cdht.py  3 4 5 300 0.3" &
xterm -hold -title "Peer 4" -e "python3 cdht.py  4 5 8 300 0.3" &
xterm -hold -title "Peer 5" -e "python3 cdht.py  5 8 10 300 0.3" &
xterm -hold -title "Peer 8" -e "python3 cdht.py  8 10 12 300 0.3" &
xterm -hold -title "Peer 10" -e "python3 cdht.py  10 12 15 300 0.3" &
xterm -hold -title "Peer 12" -e "python3 cdht.py  12 15 1 300 0.3" &
xterm -hold -title "Peer 15" -e "python3 cdht.py  15 1 3 300 0.3" &
```

Design:

This program uses two threads, which are PingThread and TCPListenThread.
PingThread is used for sending ping and receiving ping request and response from the host of each peer, and it is also used to check if the peer is still alive. In this program, the ping interval is 6 seconds and the peer is set to be killed if 5 request messages are missed.
Then TCPListenThread is used to listen to the TCP messages about requesting of files, polite depart and the message of peers who find out their successors or second successors are killed and thus ask for new successors.
Then there are many functions serving for the whole construction. There is after_input() function which is used to check whether there is client input for file requesting or quit. Then there are some functions such as file_loc(), ask_next_tcp(), receive_tcp() designed for finding location of requested file. Also, there are two functions for file transfer, receive_file() and transfer_file(). Besides, the polite depart is defined in self_depart() function that the leaving peer will send messages to its predecessors via TCP and predecessors will then receive the TCP messages and update their new successors. Finally, kill_suc() function is used by the peer who finds its first successor or second successor to ask for their successors for second successors, and this message was forwarded by TCP.
And for listening to the TCP and UDP messages, there are some signal words which indicate their identification, such as file request, ping, peer depart or request for second successors.

Message formats:
Ping message:
ping,{current peer ID},{1/2},{ping_seq_no}

The word 'ping' is to identify the ping messages among lots of UDP messages. {current peer ID} is used for its successors to locate the predecessors, {1/2} specify whether this peer is the first predecessor or the second predecessor of its target peer, and {ping_seq_no} is used for checking if its successors are still alive.

Request message:
{hash_number},{requester_ID},{filename},ask

The last word 'ask' indicates this TCP message is to ask for file transfer. Then {hash_number} is used to tell the successor the requested file's hash_number and see if this file belongs to the successor or not. {requester_ID} is used to inform the successor the requester's ID in case successor is the owner of the requested file. {filename} is used to tell the final responder that it should output the filename.

Depart message:
dapart,{current_ID},{successor_for_its_predecessors},{second_successor_for_its_predecessors }

The first word 'depart' indicates that this TCP message is for the peer to inform its predecessors that it will depart, and {current_ID} is the ID for the departing peer, {successor_for_its_predecessors} and {second_successor_for_its_predecessors} are the new successors for the predecessors.

Kill message:
my suc lost,{current_ID}
my sec suc lost,{current_ID},{sec_suc_ID}

The first sentence indicates the new successors that whether current peer's successor is lost or second successor is lost. {current_ID} informs the peer ID who requests for new successors. {sec_suc_ID} is set for those peers who lose their second successors, and they need to tell their successor its original second successor and the successor will then check if it has already updated new successors or not, and then the successor could give the right second successor back to the requested peer.

Reference:
Information about threading in network: http://www.runoob.com/python/python-multithreading.html

Demo:
https://youtu.be/Gls3uR6JbLE