Peer-to-Peer network

Class Peer

Initialised in Main given the Peer Id, its successors, predecessor and ping interval. The predecessor is set to none at initilisation.

Thread 1

Calls *send_ping* to ping first successor .

Thread 2

Calls *send_ping* to ping second successor.

Thread 3

If Peer wants to join new thread for *join_request* starts

Thread 4

Thread responsible for listening to user input in peer terminal. Deals with graceful quit, store file and request file.

Thread 5

Calls UDP listener to respond to any ping requests received.

Thread 6

Calls TCP listener to respond to quit requests, both graceful and abrupt . Also manages store requests, file requests and join requests.

Send_ping

Ping request message is encrypted with the use of the global dictionary ENCR_MTYP and sent through the UDP buffer to the indicated successor. The successor peer can then decrypt the message using MESSAGETYPE and compose a response 'pingres' for the sender to read.

> *Message_helper* is used to construct the message which if given values creates byte array messages to incorporate message type, sender Id and other extra information needed by receiver.

If the socket times out, ACK accumulation begins to determine if peer is alive. If no response is received after max accumulation which is set to 3, *lost_peer* function is called to handle quit.

Listen_UDP

Connects to UDP port to listen to any ping requests if ping request is received. It decrypts any ‘pingreq’ and encrypts response message and sends to predecessor. The predecessor value for the peer is accordingly updated and the socket closed

Listen_TCP

Binds to TCP server and listens to any incoming messages received from successors and predecessor. The data array is decrypted using MESSAGETYPE where data[0] holds the type of message received. The listener handles

- graceful quit and abrupt quit
- store request
- file request and response
- join request and response

Quit - if a quit message is received from the successor the current peer successors are updated removing the quiting peer from the network. If the current peer now becomes the last peer on the network an appropriate message is displayed other wise the updated successors are displayed on the current peer terminal.

AbortPeer - if the message type is abortPeer a message sent from the Lost_peer function . listen_TCP sends a message containing the peer′s current successor.

Store Request - once receiving a store request message from store_function listen_TCP either forwards the request to the correct peer or if the current peer id matches the correct location creates the file and accepts the store request.

File request - similar to store request the file request finds the correct peer which holds the desired file by forwarding the request to the correct peer and checking against its peer ID. This step was not entirely completed due to timing issues. Correct messages are forwarded in the system however the file itself is not copied.

File response - when the correct location is found the Peer which requested the file is given a response message ‘fileres’

Join Request - once receiving a join request from the join_function Listen TCP forwards the request onto the correct location for the new peer to join . Connection issues prevented the completion of this segment of the assignment.

File_function and store function

both perform similar tasks of determining the correct location of the file within the circular CDHT . If the correct location is not in the current Peer network a message is sent through TCP to the next successor. Due to timing issues within the network these functions firstly check to see if all predecessors having been updated before resuming.

quit_function

Called by listen_input when quit is inputted in peer terminal. The Peer waits to connect to its current predecessor and encrypts the message with the current quitting peers successors id. This allows the predecessor to accordingly update its successors and remove the quitting peer from the network.

Listen_input

Constantly listens to inputs in the terminal including - quit, store request and file request.

Join_request

Provided a join request is called this function is called which uses TCP to connect to the contact Peer aswell as a message which contains the ID of the joining peer. Listen_TCP then handles the join request however due to connection issues this was not implemented entirely .

Improvements

I had a total of 6 threads in the system and improvements in the future could be to decrease this to reduce the burden on the system. As due to this the system may lag. In future I would also aim to better organise the code to allow easy extension when progressing through the assignment. This is due to the fact that as I progressed through each step extending my previous code became more difficult and complicated making it messy and hard to follow through.A better approach would be planning each step so it can work with possible extensions. That being said possible extension for this assignment would be completing join request and file request functions. Issues while completing this assignment included address error issues which occurred due to ports not being closed properly . This prevented progressive testing and so next time I would be more wise when testing the program ensuring all ports are closed after each run.