For this project, I am using multithreads to deal with multiple sockets.
For ping message, I use UDP(datagram socket) to transfer information.
For other commands, such as ***quit*** , ***request filename*** I usd TCP socket to transfer information.

Message Design:

I put one timer and three threads into main method.
Timers sends two ping request to peer's two successors every 10 secs. And this ping request message also includes predecessor information for targeted successor, which would be useful for quit command.
And one of those three threads keep receiving UDP messages, it uses first integer data to determine whether this packet is request packet or response packet(0 is request packet 1 is response packet). If message is request packet, peer also sends a response packet to sender port.
One of those threads is in charge with sending TCP packet away, it gets input from XTerm,
then send suitable packet to targeted peers. For invalid commands, (not quit or request file), it will not send packet to any peer but print "command is not valid !!!".
The last thread is an instance of TCP server socket, it is in charge with receiving TCP packet and make right response. It uses Java pattern to read packet data, and grab information from received string then print useful data out and send response to right socket. For example, when a server socket receives a file request, it gets filename and request peer from data string and check whether it has file or not, if it does, send response to request peer, if it doesn't, transfer request to first successor.
For more information about useage, please read README.txt.

System Design :

This program has four classes,
Peer class stores all information of a user includes identity successors and predecessors(default setting is 0, but it will update after ping message received).
TCP server class has all methods about how to receive and how to deal with received packets.
It includes an algorithm about determine whether a file is in a peer or not. It is also an instance in main class, it is an independent thread, which keeps receive TCP messages.
UDPConnection class is a class with deal with ping messages. Main method calls UDP in two different threads, one is for receiving ping message, and another one is sending Ping message.
Main class class calls all other classes and gather information. It also includes a unique thread to read input and send TCP packets.

Possible improvements:

1.Maybe I can provide more details when user gives invalid commands
2. When user quit without telling predecessors, I should also let other peers detect it.(extension part which I haven't done)