1 Class and Method Design

Thread Class	Method	Function

PingClient	run( ){
	(new PingSend(s1)).start();
	(new PingSend(s2)).start();
	}
		start PingSend thread for each successor

PingSend	run( )	
	sendPing( )	build UDP connection and send PingRequest
	killed( )	if haven't received Ping Response from a successor several times, then viewed this peer is not alive anymore, send TCP request to update self successors
PingServer	run( )	after receiving Ping Request, first update self predecessors and then send back Ping Response via UDP

TCPClient	run( ){
	Scanner reader = new Scanner(system.in);
	while(cdht.terminated<2) {
	String request = reader.nextLine().toLowerCase();
	RequestSender rs = new RequestSender(request);
	rs.start();
	
	}
	reader.close();
	}
		for each input line, make it an individual TCP request thread.

RequestSender	run() {
	if(request.startsWith("request"){
	int n =Integer.parseInt(request.trim().split(" ")[1]);
	requestFile(n,id);
	}else if (request.startsWith("quit")) {
	quit();
	}
	}
		User input will have two cases, for case one which starts with "request", is a file request, goes to requestFile method.
		for case two, which is a quit request, goes to quit function

	requestFile(int fileName,String sourcePeerID)	Build TCP connection and send file request
	quit()	send successors info to predecessors and after receiving response from predecessors, close all self UDP and TCP connection
	run( )	handle all receiving Msg, and base on each msg's prefix, choose suitable method

Thread Class	Method	Function
TCPServer	handleFile( )	processing file request, if file exists there, send Response to original request peer, otherwise pass this request to next peer
	handleDeparture( )	processing departure msg from previous successors and update successors

2 Message format

PingRequest & PingResponse	self.id
File Request	“file ”+ fileName + “ source ”+ requestPeer.id
File Response	"Received a response message from peer " + self.id+”, which has the file " + fileName
Departure Request	“departure ”+ self.id +” successors ” + s1.id + “ ”+ s2.id
Departure Response	“confirm departure msg”
Request new successors(killed situation)	“successor ”+ successor_index
Response new successors(killed situation)	“newSuccessor ”+s_id

3 Other design choices

- PingRequest OutOfTime: 10 sec
- PingRequest re-send times: 3 . More than 3 times will view this successor not alive any more
- PingRequest Interval: 60 sec
- when to terminate self: use global int parameter terminated=0, once receiving departure response, terminated++, all running thread based on terminated < 2 ;
- store predecessors: use ArrayList. Only add item that are not in this list, and once list.size is 2 clear this list (make sure after updating successors, old values will be removed and new value can add in)

4 Possible Improvement

Ping interval (60 secs) is still a long time when some one is killed. Since after Ping Request Sent, one can only find out previous successors dead and update new successor. If in real world, during this long interval, someone send file request, there would be some TCP connection error.

Solutions:

- Use a shorter Ping Interval (not really solve the problem but simple)
- Auto start Ping Request Sending when TCP connection fail after several time (use a parameter X, if this TCP connection fail, X ++. When X reaches some level, start Ping Sending to detect and update successors)

5 Youtube Link

https://youtu.be/OYs9G9de3ys
(better to choose 1080p and full screen when play, since text size is quite small)