Assignment COMP3331 Computer Networks & Applications

Pedro Henrique Arruda Faustini
z5030903

University of New South Wales, Australia

May, 2015

1. Important information

Before typing "quit" in a xterm terminal, please allow enough time for both ping response messages of that screen to appear, which should take 10 seconds. And after having typed quit, please allow three ping request messages to appear, which should take 15 seconds for peers to update their new predecessors.

Also, notice that my class is called *Cdht*, not *cdht* because of Java convention, if that matters for script marking purposes.

2. Program design and how system works

The program has one major class, Cdht, which contains three inner classes: ServerUDP, ServerTCP and ClientTCP. All of these inner classes run in threads.

In the main method, a object of class Cdht is instantiated. This instance launches the three threads described above. After that, still in the main and ito an infinite loop, the peer just instantiated sends ping request messages to its successors through UDP.

The inner class ServerUDP, running "in background" in a thread, receives all UDP messages. The class ServerTCP has analogous behaviour. The class ClientTCP is always running and catching input given by the user. The input can be either a request to a file or a quit command.

In case of a request, the class sends a message to its next peer, looking for the file (the message arrives at the ServerTCP class). As described in the specification for the assignment, the message runs over the circular network until the owner of the file is found. The owner sends a message directly to the peer who made the request for that file, who then receives this message by its ServerTCP class.

In case of a quit, the peer sends two messages to its predecessors (found through the ping requests received), waits for responses and the System.exit() method is then called.

3. Message design

The messages are strings exchanged by peers. They can have different formats, according to what they mean.

- **UDP ping requests** have the format "requestUDP myPort"
- **TCP request for a file** has the format "request filename whoSentToCurrentServer whoFirstMadeTheRequest"
- **TCP quit request** has the format "update newFirstSuccessor newSecondSuccessor myPort"

- **TCP "I have the file" message** has the format "200 myPort filename"
- **TCP quit response** has the format "300 myPort"

4. Trade-offs

Since user input is made of short strings (up to 16 characters), I decided to allow five seconds of interval between each UDP request is sent. This gives enough time for someone to type "request filename" or "quit" in a terminal and see the ping messages showing up in the screen without having to wait too much.

5. Possible extensions

An extension could be the conclusion of the bonus part of the assignment, allowing the sudden exit of a peer. This would require a control of ping messages and the number of consecutive requests a peer fails to respond, as well as setting the time a peer would need to wait before considering that some response has not been received.

6. Borrowed code

In the class ServerUDP, the starting point for the code into the try/catch block was the PingServer.java file provided for the lab in week4. And the starting points for the classes ServerTCP and ClientTCP were taken from the TCPServer.java and TCPClient.java files also provided by this course. Hence, the way this program handles TCP and UDP is very analogous, changing, of course, what the program does according to the message received.