Contents

Raymonds Algorithm - A Centralized Approach

Sabbir Rashid & Rob Berman
Rensselaer Polytechnic Institute

I ABSTRACT

II INTRODUCTION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

II.I Problem Statement

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

III METHODOLOGY

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

III.I Raymond's Functions

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

III.I.i assignToken(Process p)

```
public void assignToken(Process p) {
  if ( (p.holderEnum == Process.HolderEnum.Self) && (!p.usingResource) && (!p.requestQueue.
      isEmpty()) ) {
    holderProc = p.requestQueue.pop() ;

    if (p.getProcessID() == holderProc.getProcessID()) { //i.e. the process p is at the front of its
        own queue
      p.holderEnum = Process.HolderEnum.Self;
    } else {
      p.holderEnum = Process.HolderEnum.Neighbor;
      holderProc.holderEnum = Process.HolderEnum.Self ;
    }

    p.asked = false;

    if (p.holderEnum == Process.HolderEnum.Self) {
      p.usingResource = true;
    } else {
      assignToken(holderProc); // Check this, supposed to be "send token to holder"
    }
  }
}
```

III.I.ii sendRequest(Process p)

```
public void sendRequest(Process p) {
  if ( (p.holderEnum != Process.HolderEnum.Self) && (!p.requestQueue.isEmpty()) && (!p.asked) ) {
    sendRequest(holderProc);
    p.asked = true;
  }
}
```

III.I.iii requestResource(Process p)

```
public void requestResource(Process p) {
  p.requestQueue.push(p);
  assignToken(p);
  sendRequest(p);
}
```

III.I.iv releaseResource(Process p)

```
public void releaseResource(Process p) {
  p.usingResource = false;
  assignToken(p);
  sendRequest(p);
}
```

III.I.v receivedRequestFromNeighbor(Process p, Process neighbor)

```
public void receivedRequestFromNeighbor(Process p, Process neighbor) {
  p.requestQueue.push(neighbor);
  assignToken(p);
  sendRequest(p);
}
```

III.I.vi receivedToken(Process p)

```
public void receivedToken(Process p) {
  p.holderEnum = Process.HolderEnum.Self ;
  holderProc = p;
  assignToken(p);
  sendRequest(p);
}
```

III.II Client Code

```
package sockets;

import java.io.*;
import java.net.*;


public class Client {
        public static void main(String[] args) throws IOException {

        if (args.length != 2) {
            System.err.println(
                "Usage: java Client <host name> <port number>");
            System.exit(1);
        }

        String hostName = args[0];
        int portNumber = Integer.parseInt(args[1]);
        System.out.println("CLIENT: About to try to create Client Socket");
        try (
            Socket clientSocket = new Socket(hostName, portNumber);
                PrintWriter out =
                new PrintWriter(clientSocket.getOutputStream(), true);
                BufferedReader in =
                new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));
            BufferedReader stdIn =
                new BufferedReader(
                    new InputStreamReader(System.in));
                ){
                String userInput;
                System.out.println("CLIENT: About to wait for user input.");
                System.out.println("Select the following command that you want to execute:");
                System.out.println("1: create <filename>: creates an empty file named <filename>");
                System.out.println("2: delete <filename>: deletes file named <filename>");
                System.out.println("3: read <filename>: displays the contents of <filename>");
                System.out.println("4: append <filename> <line>: appends a <line> to <filename>");
                System.out.println("5: exit: exits the program");

                while ((userInput = stdIn.readLine()) != null) {
                        out.println(userInput);
                        System.out.println(in.readLine());
                        if (userInput.contains("read")){
                                String ans = "";
                                while(in.ready())
                    {
                                        ans=in.readLine();
                        System.out.println("CLIENT: In inner while loop.");
                        System.out.println(ans);
                    }
                    System.out.println("CLIENT: Exited inner while loop.");
                        }
                }
            System.out.println("CLIENT: Exited while loop.");
            clientSocket.close();
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host " + hostName);
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to " +
                hostName);
            System.exit(1);
        }
    }
}
```

III.III Multithreaded Server Code

```
package sockets;

import java.io.BufferedReader;
import java.io.File;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

import main.Main;

public class MultiThread {
        public static void main(String[] args) throws Exception {
                if (args.length != 1) {
                        System.err.println("Usage: java Server <port number>");
                        System.exit(1);
                }

                int portNumber = Integer.parseInt(args[0]);

                @SuppressWarnings("resource")
                ServerSocket m_ServerSocket = new ServerSocket(portNumber);

                int id = 0;
                while (true) {
                        Socket clientSocket = m_ServerSocket.accept();
                        ClientServiceThread cliThread = new ClientServiceThread(clientSocket, id++);
                        cliThread.start();
                }
        }
}

class ClientServiceThread extends Thread {
        Socket clientSocket;
        int clientID = -1;
        boolean running = true;

        ClientServiceThread(Socket s, int i) {
                clientSocket = s;
                clientID = i;
        }

        public void run() {
                System.out.println("Accepted Client : ID - " + clientID + " : Address - "
                                + clientSocket.getInetAddress().getHostName());
                try {
                        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.
                             getInputStream()));
                        System.out.println("SERVER: Created buffered reader in.");
                        PrintWriter out = new PrintWriter(new OutputStreamWriter(clientSocket.
                             getOutputStream()),true);
                        System.out.println("SERVER: Created print writer out.");

                        while (running) {
                                System.out.println("SERVER: In running loop.");
                                //String result = console.nextLine();
                                //String result = inputLine;
                                String result = in.readLine();
                                //Note: Calling create, delete, read, and append go here:
                                File testFile = null;
                                if(result.substring(0,6).equalsIgnoreCase("create"))
                                {
                                        out.println("Creating File ... ");
                                        testFile = Main.CreateFile(result.substring(7, result.length()));
                                }
                                else if(result.substring(0,6).equalsIgnoreCase("delete"))
                                {
                                        out.println("Deleting File ... ");
                                        Main.DeleteFile(result.substring(7, result.length()));
                                }
                                else if(result.substring(0,4).equalsIgnoreCase("read"))
                                {
                                        String temp = Main.ReadFile(result.substring(5,result.length()))
                                            ;
                                        out.println("Reading File...\n" + temp);
                                        out.flush();
                                }
                                else if(result.substring(0,6).equalsIgnoreCase("append"))
                                {
                                        out.println("Appending to File...");
                                        String tmp = result.substring(7, result.length());
                                        int index = tmp.indexOf(' ');
                                        Main.AppendFile(tmp.substring(0,index),tmp.substring(index+1,
                                            tmp.length()));
                                }
                                else if(result.substring(0,4).equalsIgnoreCase("exit"))
                                {
                                        out.println("Exiting ... ");
                                        out.flush();
                                        running = false;
                                        System.out.print("Stopping client thread for client : " +
                                            clientID);
                                        Main.ExitConnection();
                                }
                                else
                                        out.println("Error: Invalid Command");

                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

IV MAIN - READ, WRITE, APPEND, DELETE

```
package main;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import raymonds.Process;

public class Main {

        public static void main(String[] args) throws IOException {
                FileReader fr = new FileReader("tree.txt");
                String input = br.readLine();
                boolean first = true;
                ArrayList<Process> processes = new ArrayList<Process>();
                while(input!=null)
                {
                        if( first )
                        {
                                processes.add(new Process(input.substring(1, 2),Process.HolderEnum.
                                    Neighbor,false,false));
                                processes.get(0).addNeighbor(new Process(input.substring(3, 4),Process.
                                    HolderEnum.Neighbor,false,false));
                                processes.add(new Process(input.substring(3, 4),Process.HolderEnum.
                                    Neighbor,false,false));
                                processes.get(1).addNeighbor(new Process(input.substring(1, 2),Process.
                                    HolderEnum.Neighbor,false,false));
                                first = false;
                        }
                        else
                        {
                                int index = 0;
                                boolean found=false;
                                for(int i=0;i<processes.size();i++)
                                {
                                        if(input.substring(1,2).equals(processes.get(i).getProcessID()))
                                        {
                                                found=true;
                                                index=i;
                                        }

                                }
                                if (!found)
```

```
{
        processes.add(new Process(input.substring(1, 2),Process.
            HolderEnum.Neighbor,false,false));
        processes.get(processes.size()-1).addNeighbor(new Process(
            input.substring(3, 4),Process.HolderEnum.Neighbor,false,
            false));
}
else
{
        if (!processes.get(index).getNeighbors().contains(new Process(
            input.substring(3, 4),Process.HolderEnum.Neighbor,false,
            false)))
        {
                processes.get(index).addNeighbor(new Process(input.
                    substring(3, 4),Process.HolderEnum.Neighbor,false,
                    false));
                System.out.println("CONTAINS 1");
        }
}
found=false;
index = 0;
for(int i=0;i<processes.size();i++)
{
        if(input.substring(3,4).equals(processes.get(i).getProcessID()))
        {
                found=true;
                index=i;
        }
}
if (!found)
{
        processes.add(new Process(input.substring(3, 4),Process.
            HolderEnum.Neighbor,false,false));
        processes.get(processes.size()-1).addNeighbor(new Process(
            input.substring(1, 2),Process.HolderEnum.Neighbor,false,
            false));
}
else
{
        if (!processes.get(index).getNeighbors().contains(new Process(
            input.substring(1, 2),Process.HolderEnum.Neighbor,false,
            false)))
        {
                processes.get(index).addNeighbor(new Process(input.
                    substring(1, 2),Process.HolderEnum.Neighbor,false,
                    false));
                System.out.println("CONTAINS 2");
        }
}
```

```
                }
                input=br.readLine();
            }
            for(int i=0;i<processes.size();i++)
            {
                System.out.println(processes.get(i).getProcessID());
                System.out.print("Neighbors: ");
                for(int j=0;j<processes.get(i).getNeighbors().size();j++)
                    System.out.print(processes.get(i).getNeighbors().get(j).getProcessID()+
                      " ");
                System.out.println();
            }
        }

        public static File CreateFile( String fileName) throws IOException {

            File file = new File(fileName);
            file.createNewFile();
            return file;
        }

        public static void AppendFile( String fileName, String line) throws IOException {

            FileWriter writer = new FileWriter(fileName, true);
            writer.append(line + "\n");
            writer.flush();
            writer.close();

        }

        public static String ReadFile( String fileName) throws IOException {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));
            String input = reader.readLine();
            String result = input;
            while(input!=null)
            {
                input=reader.readLine();
                if(input!=null)
                    result = result + "\n"+ input;
            }
            reader.close();
            return result;
        }

        public static void DeleteFile( String fileName) throws IOException {
            Runtime.getRuntime().exec(new String[]{"bash","-c","rm " + fileName});
        }

        public static void ExitConnection()
        {

        }

}
```

V CONCLUSIONS

Quisque consectetuer. In suscipit mauris a dolor pellentesque consectetuer. Mauris convallis neque non erat. In lacinia. Pellentesque leo eros, sagittis quis, fermentum quis, tincidunt ut, sapien. Maecenas sem. Curabitur eros odio, interdum eu, feugiat eu, porta ac, nisl. Curabitur nunc. Etiam fermentum convallis velit. Pellentesque laoreet lacus. Quisque sed elit. Nam quis tellus. Aliquam tellus arcu, adipiscing non, tincidunt eleifend, adipiscing quis, augue. Vivamus elementum placerat enim. Suspendisse ut tortor. Integer faucibus adipiscing felis. Aenean consectetuer mattis lectus. Morbi malesuada faucibus dolor. Nam lacus. Etiam arcu libero, malesuada vitae, aliquam vitae, blandit tristique, nisl.