Fall 12-2010

Online Application Monitoring Tool

Sathya Anandan
San Jose State University

Online Application Monitoring Tool

A Project
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science
By
Sathya Anandan
Dec 15, 2010

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

ONLINE APPLICATION MONITORING TOOL
by
Sathya Anandan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp	Department of Computer Science	Date
Dr. Agustin Araya	Department of Computer Science	Date
Mr. Elangovan Kulandaivelu	Apple Inc.,	Date

APPROVED FOR THE UNIVERSITY

Associate Dean	Office of Graduate Studies and Research	Date

ABSTRACT

In some classes, students take online tests and some types of network activity (for example trying to find the answers in www.google.com) will be considered as cheating during the exam. It would be useful if instructor could monitor online activities of each student to detect cheating. The goal of this project is to develop such a tool using client/server architecture. Tool should display the student's hostname and the website visited during unauthorized web activity. Professor should able to see the applications opened by all students and he will have an option to view the log files of all the students during the session and also at the end of the session. The tool will display the student's hostname during login and logout from the network.

ACKNOWLEDGEMENTS

I would like to thank my project advisor Dr. Mark Stamp for his guidance and insights throughout the project. I would also like to thank my committee member Dr. Agustin Araya for providing me with his valuable feedback.

I specially like to thank my husband Guna, parents, and brother for their encouragement and motivation throughout the Masters program.

Table of Contents

1. Introduction 1
2. Monitoring Tools 2
2.1. Wireshark 2
2.1.1. Purposes of Wireshark 2
2.1.2. Features of Wireshark 3
2.1.3 Wireshark does not do the following 3
2.1.4 Snapshot of Wireshark 3
2.1.5. Remote Capture Interfaces: 4
2.1.6. Drawbacks of Wireshark 7
2.2. Kismet 7
2.3. Developing Our Own Tool 8
3. High Level Design 8
4. Requirements 10
4.1. Wireless USB Router 10
4.2. Student Requirements 10
4.3. Professor Requirements 11
5. Explanation of the Tools 12
5.1. Client Program 12
5.2. Server Program 14
5.2.1. Manual Refresh of Client Machines 14
5.2.1. View Applications Opened by Students 15
5.2.3. Black List GUI 16
5.2.4. White List GUI 17
5.2.5. Log File 18
5.2.9. Pop-up for Black Listed Website 23
5.2.10. Pop-up for Whitelisted Websites 24
5.2.11. Exit 25
6. Possible Attacks 26
7. Solution 27
7.1. Auto Refresh 27
8. Testing 29

9. Conclusions and Future Work..... 32
10. References..... 33

List of Figures:

Figure 1: Snapshot of Wireshark packet capturing. 4
Figure 2: In remote machine rpcapd.exe is running. 5
Figure 3: Machine A gives IP address of Machine B 6
Figure 4: Machine B opens Google and the packet is captured in Machine A 7
Figure 5:General Block Diagram. 8
Figure 6: User Interface of the Monitoring Tool. 9
Figure 7: Running clients are shown in the monitoring tool. 14
Figure 8: Professor Viewing the Application Details of Particular Student 15
Figure 9: BlackList to update the header of the applications and website. 17
Figure 10: White List to update the header of the applications and website. 18
Figure 11: Log List during the session once clicked on the View log File. 19
Figure 12: Professor selects particular client machine. 20
Figure 13: Log list after the session. 21
Figure 14: Screenshot for the BlackList Website pop-ups 24
Figure 15: Screenshot for the Whitelisted Website Pop-ups 25
Figure 16: Pop-up to show the Student Logged In 28
Figure 17: Pop-up to show the Student Logged Out. 28

List of Tables:

Table 1: Activities performed by students. 31
Table 2: Activities performed by professor. 32

1. Introduction

Today, in many university classes, students take some tests during class hours using their own laptop computers. Often, it is necessary for students to have Internet access to, for example, obtain the test paper, access test-specific online resources, and turn in the completed test. There are many ways that students can attempt to cheat on such a test. For example, a student can visit websites to search for answers, or a student can chat online with other students or friends to discuss possible solutions.

There is no easy way for a professor to detect such cheating. When a professor approaches a cheating student, the student can easily close a window to hide unauthorized activity. A professor could randomly spot-check student laptops during the test, but this is likely to be disruptive and might miss many cases of cheating.

In this paper, we propose a solution for detecting cheating under such a scenario. We have developed a tool that alerts the professor whenever a student visits a forbidden website or performs other unauthorized online activity. Using our tool, the professor can specify a black list and/or a white list. In the black list, the professor will have a list of websites that students should not visit during the exam. In contrast, the white list contains website addresses that the students are specifically allowed to visit during the exam. For example, students might be allowed to access the computer science department website to download the test and to upload their solutions, but Google might be strictly off limits. The tool monitors the students' machines and gives the professor a warning message when the student is trying to perform some unauthorized activity. A warning is also provided if a student disconnects from the wireless access point, which prevents someone from simply using a different access point for cheating. The tool also displays all the applications and websites accessed by the students.

Next, we discuss Wireshark [2.1] and Kismet [2.2], which are two online monitoring tools. We mention some of the problems inherent in attempting to use these tools in classroom monitoring. Then we discuss the design, development, and implementation of our new tool. Then we discuss

some experimental results obtained when testing the tool in a classroom setting. Finally, we explore possible weaknesses in our approach by considering specific attacks that can be performed by students. Finally, we consider potential future work that might further improve our online monitoring tool.

2. Monitoring Tools

2.1. Wireshark

Wireshark is a network packet analyzer which will seek to capture network packets and will display the packets in detail, such as packet number, protocol name, source address, and destination address [1]. In past decades, these kinds of tools were very expensive but nowadays the Wireshark is available as an open source [1].

By using the IP address of student computers, an instructor can monitor the online activity through a Wireless Access Point (WAP). For the students to access the internet, the wireless network should be connected to a wired network via a WAP [2].

Our assumption is that students and the instructor will be connected to the internet through the same WAP [4]. Each student will have a different IP address for their system [4]. Wireshark must be installed on the instructor's computer and he must know the IP addresses of each student's computer. The instructor can then filter the IP addresses of his students' computers in the monitored IP traffic using Wireshark and find out whether they are cheating or not based on the IP traffic[2].

2.1.1. Purposes of Wireshark

Wireshark is used for various purposes by engineers, developers, and network administrators. Some examples of Wireshark use are as follows.

- Troubleshoot network problems by network administrators.
- Observe security problems by network security engineers.
- Debug protocol implementations by developers.
- Study network protocol internals by individuals.

2.1.2. Features of Wireshark

Features provided by Wireshark are as follows.

- Live network packet has been captured from the network interface.
- Each and every packet has been displayed with the detailed protocol information.
- Packets that have been captured can be saved and opened later. Filter packets based on specific criteria. For example, filter packets based on the protocol.
- Colorize the packet display based on filters.

2.1.3 Wireshark does not do the following

- Wireshark is not an Intrusion Detection System [1]. For example, if someone changes network activity or do something to the network which they are not allowed to do, then Wirehsark will not give any alert message.
- Wireshark does not manipulate things over the network. For example, it does not send packets over the network.

2.1.4 Snapshot of Wireshark

A snapshot of Wireshark Live Packet Capturing is shown in Figure1.

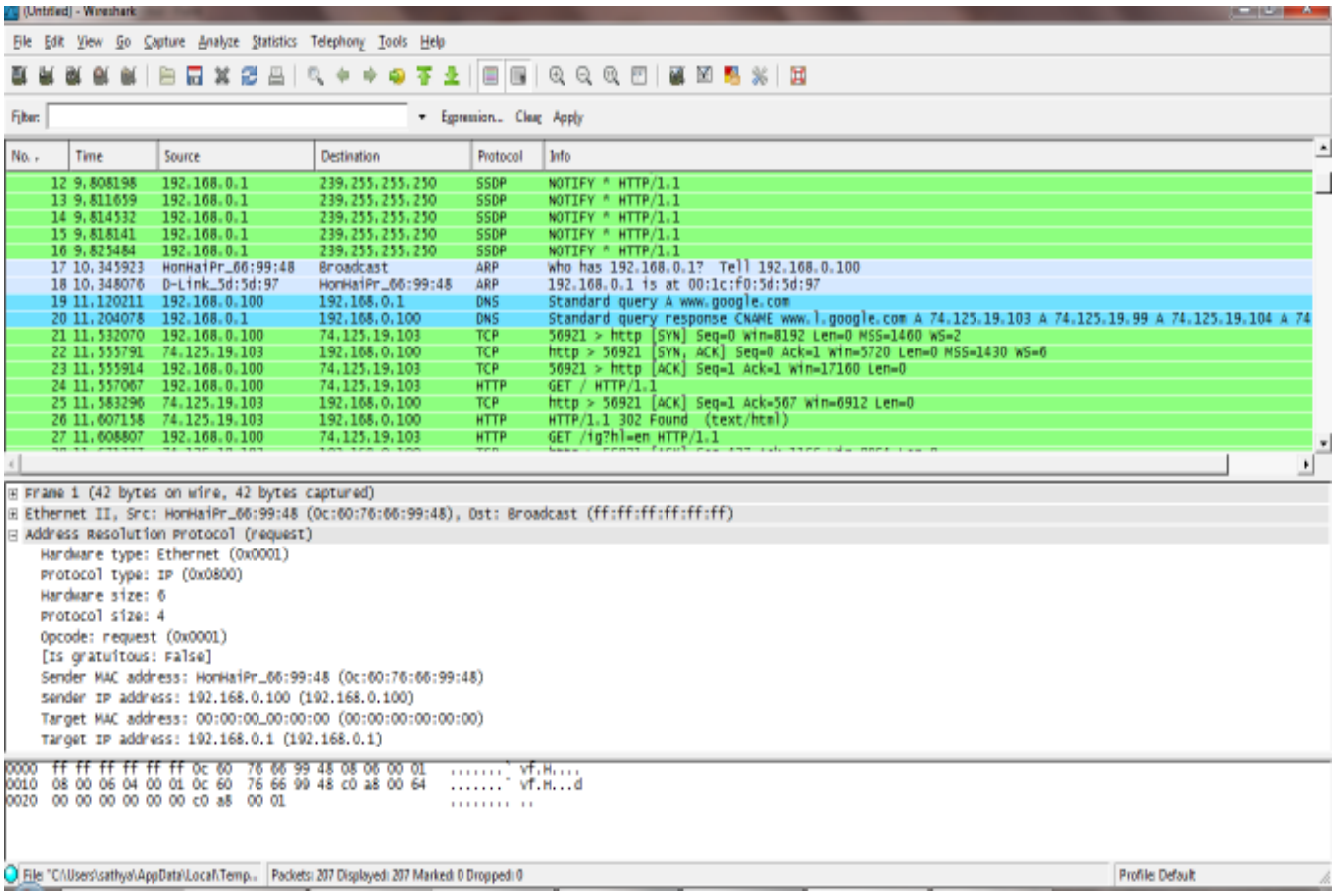

Figure 1: Snapshot of Wireshark packet capturing.

2.1.5. Remote Capture Interfaces:

Wireshark can capture remote packet data [1]. One of the major requirements for the remote capture interface is that the target machine Remote Capture Protocol (rcapd.exe) service must be running. Remote Capture Protocol must be started from the control panel [1].

For example, suppose there are three machines, Machine A, Machine B, Machine C. All three A, B, and C are connected to the same network. Machine A will have Wireshark running and Machine B will have WinPcap and rpcapd.exe files running, as shown in Figure 2. The machines have the following IP addresses.

Machine A IP address 192.168.0.100

Machine B with WinPcap and rpcapd.exe IP address 192.168.0.101

Machine C IP address 192.168.0.102

Gateway for all three machines 192.168.0.1

Now Machine A will give the IP address in the remote interface of Machine B in which rpcapd.exe is running, as shown in Figure 3.

Machine B opens www.google.com then, in Machine A, the Wireshark captures the packets and the snapshot is shown in Figure 4. Once Machine A has been connected to the network using one remote machine it can see all the machines connected to the same network. Machine C opens www.google.com and the packet is captured by Machine A, as shown in Figure 5.

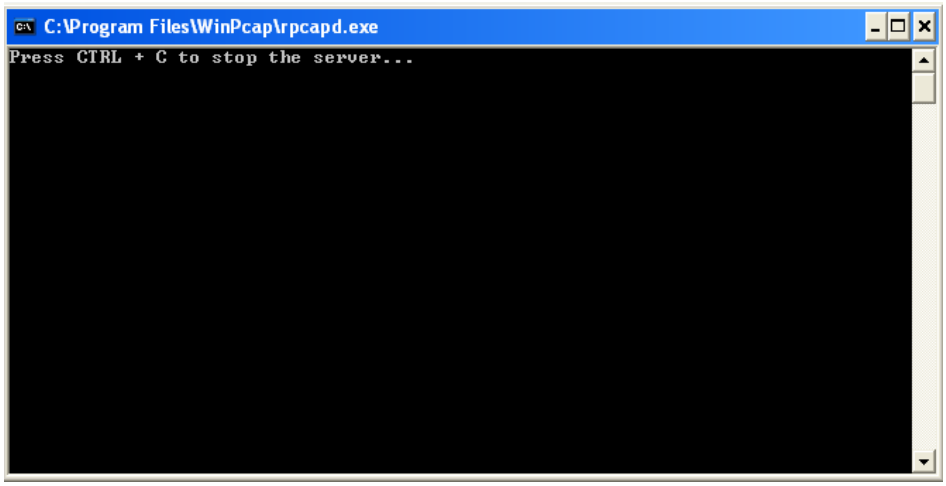

Figure 2: In remote machine rpcapd.exe is running.

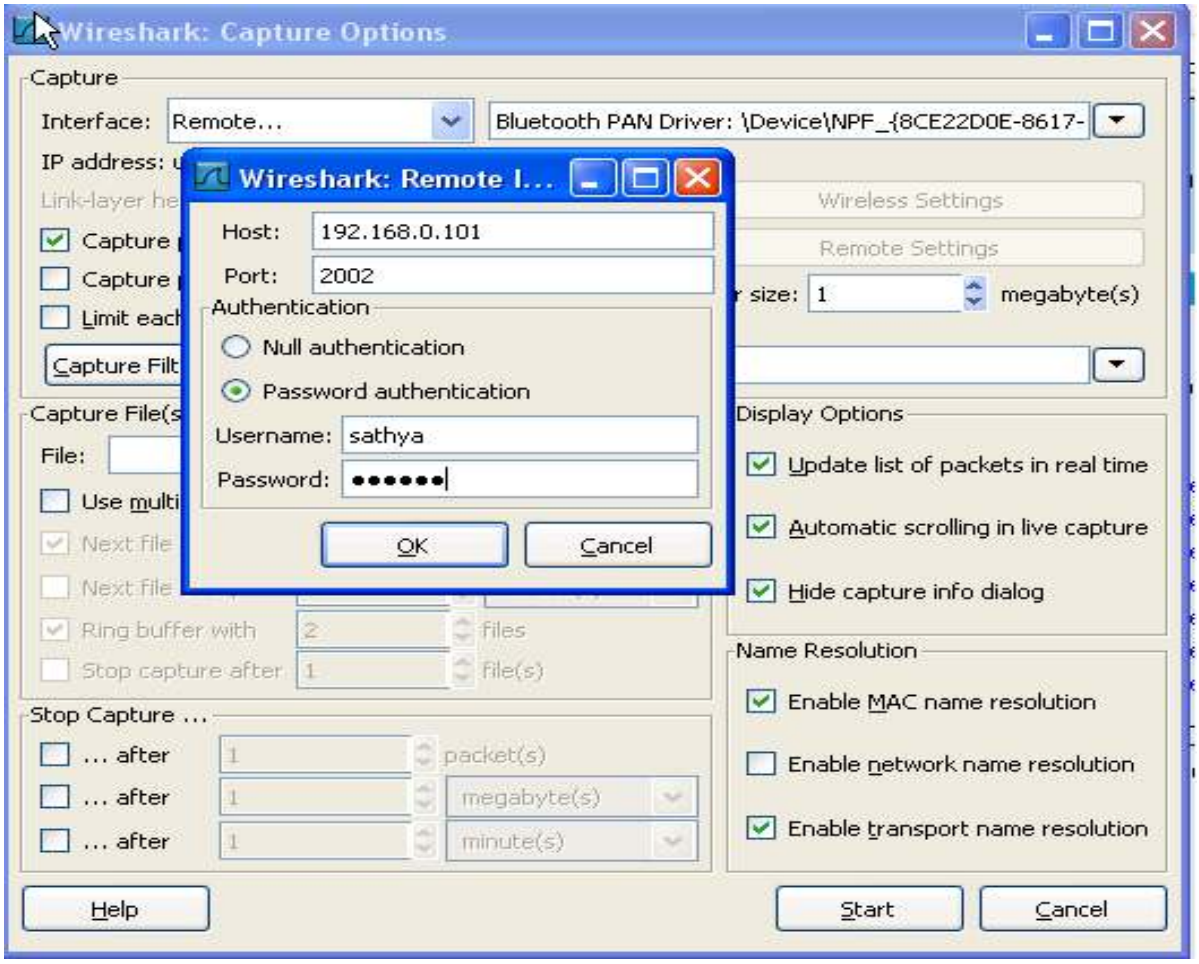

Figure 3: Machine A gives IP address of Machine B

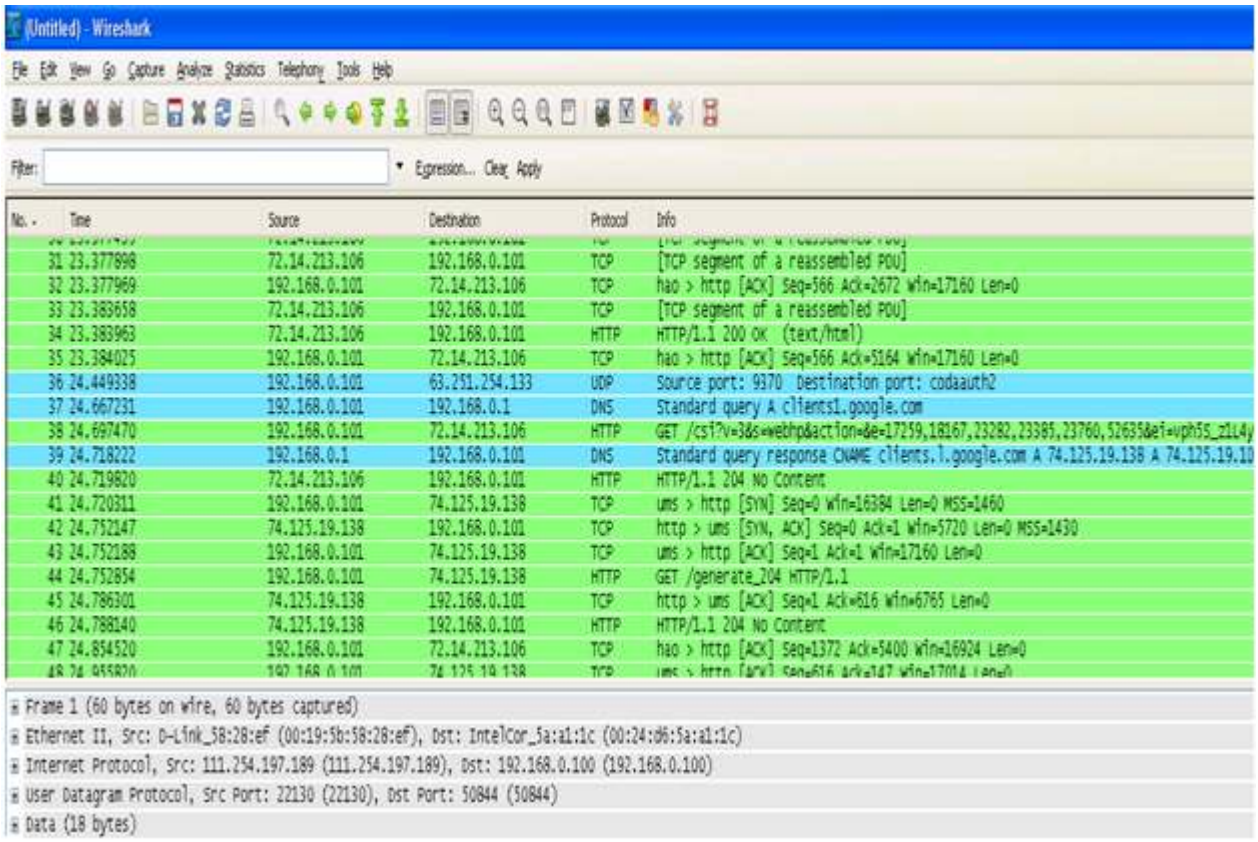

Figure 4: Machine B opens Google and the packet is captured in Machine A

2.1.6. Drawbacks of Wireshark

Wireshark captures the network packets and will display them, but it does have the following disadvantages:

- Wireshark is an open source tool, so it is hard to implement and integrate with our own plug-in.
- Wireshark is not user-friendly in our application because it is difficult to keep track of the activity of every student.

2.2. Kismet

Kismet is a 802.11 (802.11a, 802.11b, 802.11g, 802.11n) wireless detector, packet sniffer, and Intrusion Detection System. Kismet can be used to work with any wireless card [5].

For layer 2 and layer 3 attacks Kismet provides stateful and stateless IDs. An advantage of Kismet is that it costs nothing. Disadvantages of Kismet include an interface that is not user friendly [15] and difficult to implement and integrate with our own plug-in.

2.3. Developing Our Own Tool

We have developed an online monitoring tool using client/server architecture [6] in Java. The goal of this tool is to identify cheating students in a manner that is easily managed by the professor.

3. High Level Design

A general block diagram describing the activities performed by students, professor, and tool is shown in the below figure.

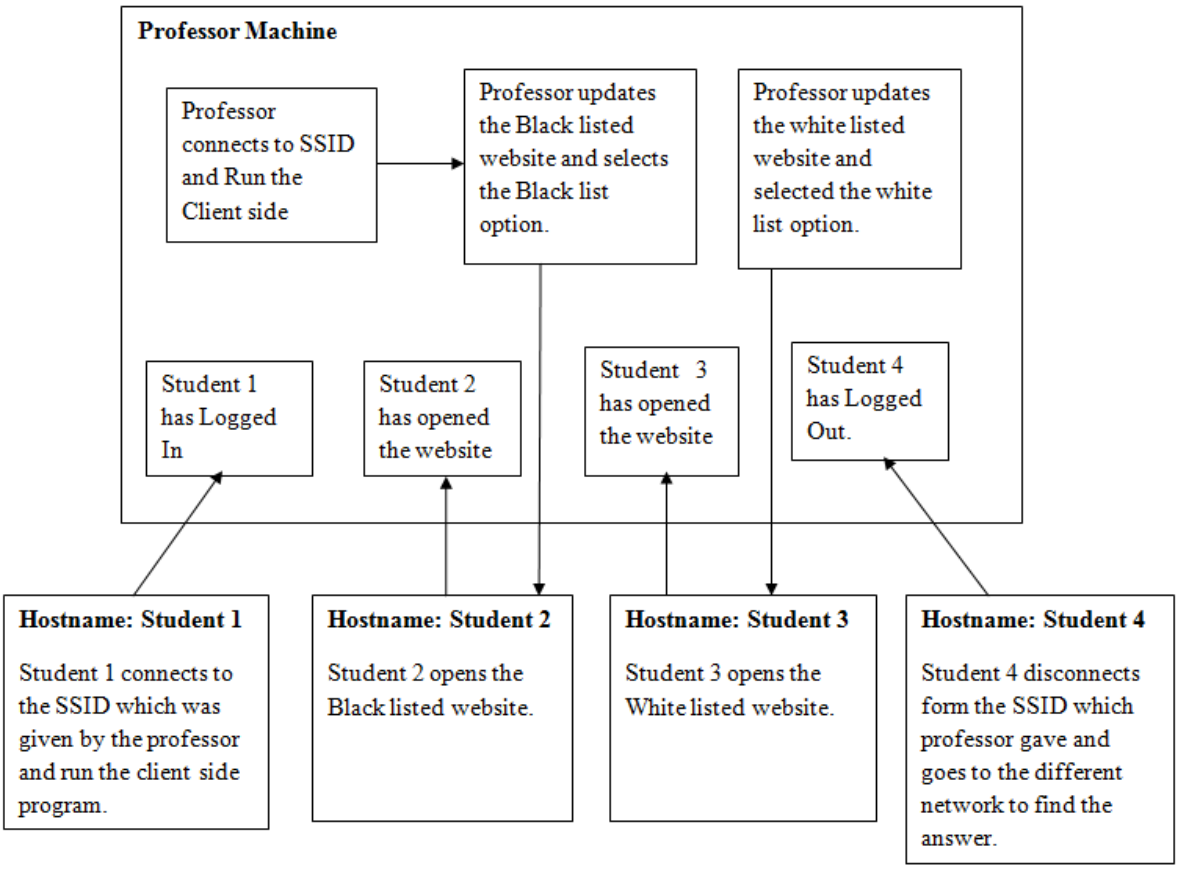

Figure 5:General Block Diagram.

Our tool will have separate programs for the students and the professor. Students will install the client-side program (for example batch program) and they will execute the program after they are connected to the network [9]. The program invokes the remote machines. The monitoring tool is installed in the instructor's machine. Once the instructor is connected to the network, he will

execute the tool. The instructor will be able to see student machines that are connected to the network with their host name. When connecting to the network for the first time, students will need to supply the instructor with the host name for identification. The user interface of the tool is shown in the Figure 6.

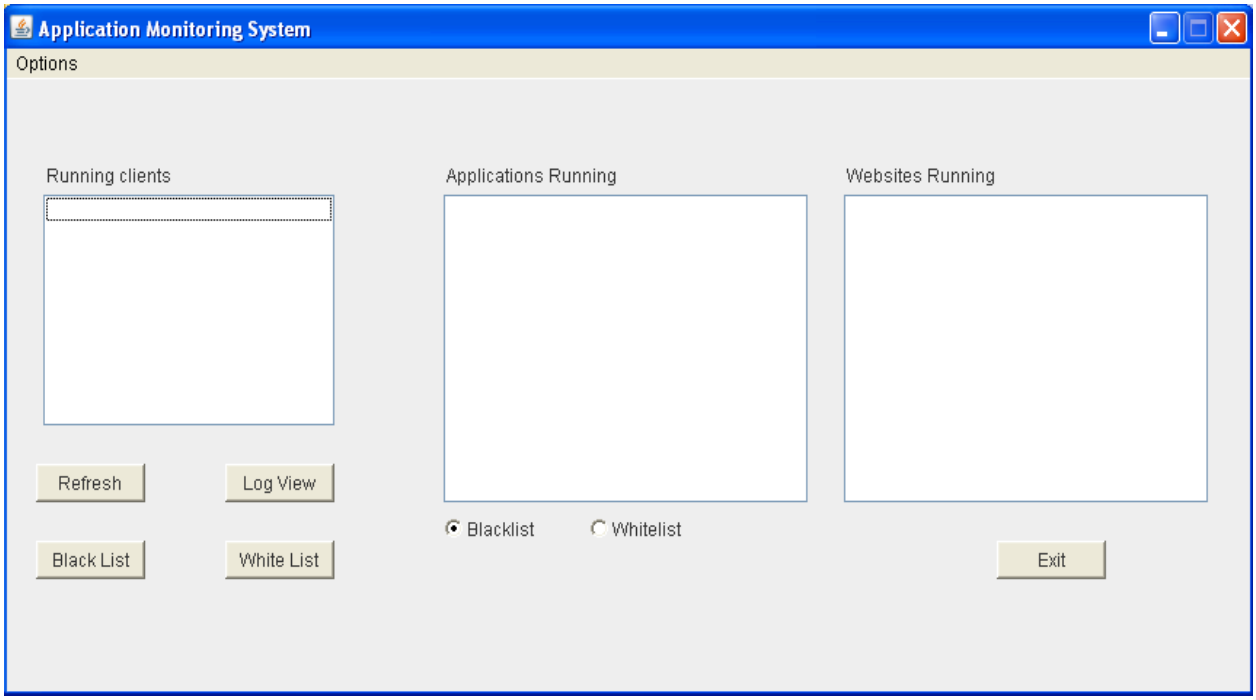

Figure 6: User Interface of the Monitoring Tool.

Our tool will have a user interface with the client running (student's host name) connected to a particular class SSID supplied by the professor. The professor can view the applications running on each individual student's machine. The professor has the option to add or update the black listed websites and white listed website.

The professor can view student activities from earlier in the session using the View Log File. The View Log File displays the student hostnames and the professor can access a given student's activity by clicking on their hostname. There is a Refresh option which allows the instructor to refresh and see current activity, for instance, if someone has entered into the network or if someone has left the network.

The professor can select either black listed or white listed sites. When the tool is started it will have “Blacklisted” selected by default. If any student visits a black listed website a pop-up will open in the professor’s machine with the student’s name and the black listed website name. If the instructor selects the “Whitelisted” option then a pop-up will open in the instructor’s machine with the student’s name and the non-white listed website name..

A log file will be created for each session showing the student’s network activity, and it will be saved in the professor’s machine. If the instructor wants to see all network activity of a particular student after the session is over, then he can view the log file at any time [18].

4. Requirements

4.1. Wireless USB Router

The client and server machines need to be connected to the same gateway. In order to connect to the same gateway, we are using a wireless USB router [7] called Windy31. We plug the Windy31 into the professor’s machine, which is connected to the internet. We create an SSID [20] and password for our wireless USB router and ask the students to connect to the same SSID using the provided password [16].

4.2. Student Requirements

In order to run the tool successfully students should have the following requirements.

Requirement 1:

Students should have Windows OS.

Requirement 2:

Students will need to connect to the SSID which was given by the professor. Once they are connected, they should type the “Net View” [8] command in their command prompt. Students should be able to see the host name of their computer and other computers that are connected to the network.

The following steps should be followed by the students to enable file sharing:

1. Turn off Windows Firewall during class hours.

2. If antivirus software prevents sharing, it will need to be turned off during class time. For example, turn off the Firewall in MacAfee Antivirus.

3. In Network Places turn on the option of file sharing within the network.

4. The WorkGroup of your computer has to be MSHOME. To verify this, right click on My Computer and click on Properties. If the WorkGroup is not MSHOME then edit the WorkGroup. Also make sure your computer name is in your name or is otherwise easily identifiable as yours.

5. Now type the "Net View" command at the command prompt.

Requirement 3

Students should have Java installed on their machine.

Requirement 4

Students should run the program given to them by the professor.

4.3. Professor Requirements

The professor should follow the requirements given below in order to run the tool in his machine and monitor the student machines.

Requirement 1

Professor should be running Windows OS.

Requirement 2

The professor will connect to the internet using Windy31 and create the SSID and password to give to the students.

Requirement 3

Type the "Net View" command and make sure you can see your host name and all student host names. The steps below should be followed to ensure that students and professor are connected to the same WorkGroup and to enable file sharing.

1. Windows Firewall will need to be disabled during class hours.

2. If antivirus software is preventing the option of sharing to the network, please it turn it off during class time. For example, turn off the Firwall in MacAfee Antivirus.

3. In the Network and Sharing Center "turn on" the option of file sharing within the network [14].

4. WorkGroup has to be MSHOME. To verify this, right click on My Computer and click on Properties. If the WorkGroup is not MSHOME then edit the WorkGroup [14]. Also make sure your computer name is in your name or is easily identifiable as belonging to you.

5. Type the “Net View” command in the command prompt.

Requirement 4

The professor should have Java installed in the machine.

Requirement 5

The professor should have Microsoft Office 2007 installed in the machine.

Requirement 6

Run the server side program.

5. Explanation of the Tools

5.1. Client Program

In order to get the applications running on the client machines we use the getApplication method to retrieve the information. We use tasklist.exe to get the running processes of the client machine. This getApplication method retrieves a list of all the applications and their header names running on the student’s machine. It keeps all the header names in an array list and returns to the caller of this method whenever needed.

```
public ArrayList getApplication()
 {
 ArrayList al=new ArrayList();
 try {
 Process p = Runtime.getRuntime().exec("tasklist.exe /v /FO LIST");
 BufferedReader in= new BufferedReader(new InputStreamReader(p.getInputStream()));
 String str = in.readLine();
 while (str!=null) {
 if (str.startsWith("Window Title:"))
     {
          String appName = str.substring(13).trim();
          if(!appName.equals("N/A"))
         al.add(appName);
           }
         str = in.readLine();
          }
         } catch (IOException e) {
         e.printStackTrace();
         }
return al;
}
```

We also create the RMI registry in the main method by using createRegistry with port no 1099 and bind with the client class. So when a student executes the client program, everything will be done in an instant.

```
public static void main (String[] args) throws Exception
{
          // Bind the remote object's stub in the registry
        Registry registry = LocateRegistry.createRegistry(1099);
        System.out.println("Started....");
        registry.bind("Client",new Client());
}
```

In order to make client class remote class, client program implements the following.

```
public interface  IClient extends java.rmi.Remote
{
   public ArrayList getApplication() throws java.rmi.RemoteException;
}
```

5.2. Server Program

In server program, we have to see all the client machines that are connected to the network. To do this we use the "Net View" command to retrieve a list of all computers connected to the gateway.

```
Process p=r.exec("net view");
```

In order to get the list of client machines that executes the client program we use Naming.lookup method to get the remote object of the client machine. If a client machine is connected to the same gateway and running RMI, it will be stored in the clientList array. If the client machines are connected to the same gateway but not running the RMI, nothing will be stored.

```
iClient=(IClient)Naming.lookup("rmi://"+str+"/Client");
clientList.add(str);
```

Once we get the client list, all client machines will be listed in our tool.

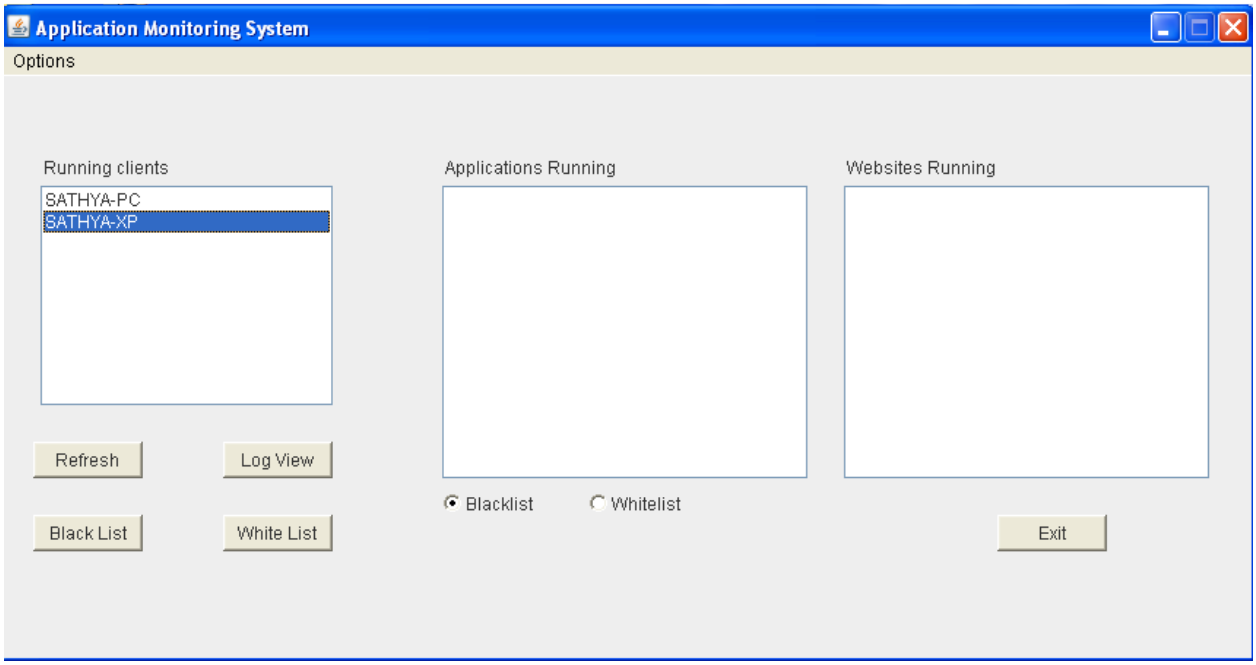

Figure 7: Running clients are shown in the monitoring tool.

5.2.1. Manual Refresh of Client Machines

In our tool we have the Refresh command button, which is used to refresh the clients running. This is the manual refresh, which is used by the professor when he wants to see the updated client list.

5.2.1. View Applications Opened by Students

If a professor wants to see what applications are opened by a particular student then, once the client list has been created, he can double click on the particular client machine's name. In order to get the applications running from the particular client machine, we use getApplication with one parameter. This method is called when the professor double clicks on the particular student machine's name. When we pass the host name, this method gets the list of all application headers running in that particular machine.

```
public  ArrayList getApplication(String s) throws Exception{
    clientList.removeAll(clientList);
    clientList=getClients();
            try{
                iClient=(IClient)Naming.lookup("rmi://"+s+"/Client");
                clientApplicationList=iClient.getApplication();
                 bList=null; }
                        catch(Exception e)
                        { e.printStackTrace();}
return clientApplicationList;  }
```

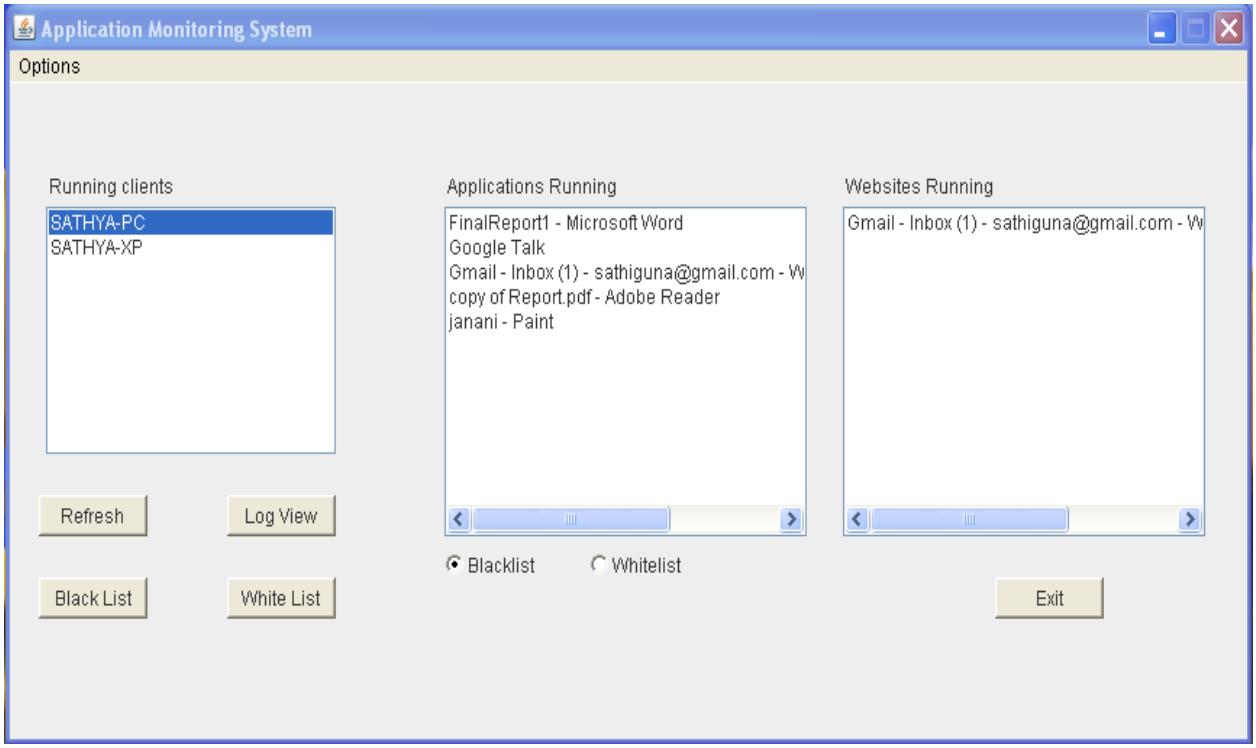

Figure 8: Professor Viewing the Application Details of Particular Student

Application Running will have all the applications and websites running and Website Running will show only the websites that are running. When a professor is viewing the applications opened by a student, it should refresh each time. For example every 10 seconds the Application Running must be refreshed otherwise it would still be showing the same applications as when professor clicked on the particular client (student) machine name. The listRefresh() method is used to automaticly refresh the applications. We call getItem method of clientList to get the client (student) name selected by the professor. We use the getApplication method to get the list of application names running in the selected client, in order to pass in the parameter. The following code is used to auto refresh the applications running on the selected client (student) machine.

```
public void listRefresh(){
  try {
String s=clientList.getItem(clientList.getSelectedIndex());
clientApplicationList.removeAll(clientApplicationList);
applicationList.removeAll();
websiteList.removeAll();
clientApplicationList=server.getApplication(s);
for (int j = 0; j<clientApplicationList.size(); j++){
apps=clientApplicationList.get(j).toString();
applicationList.add(apps);
websiteList.add(apps); }
catch(Exception e)
{ e.printStackTrace(); } }
```

5.2.3. Black List GUI

In our tool we use the Blacklist command button to update the black list. Once we click on the Blacklist command button the small frame will be invoked, as shown in Figure 9.

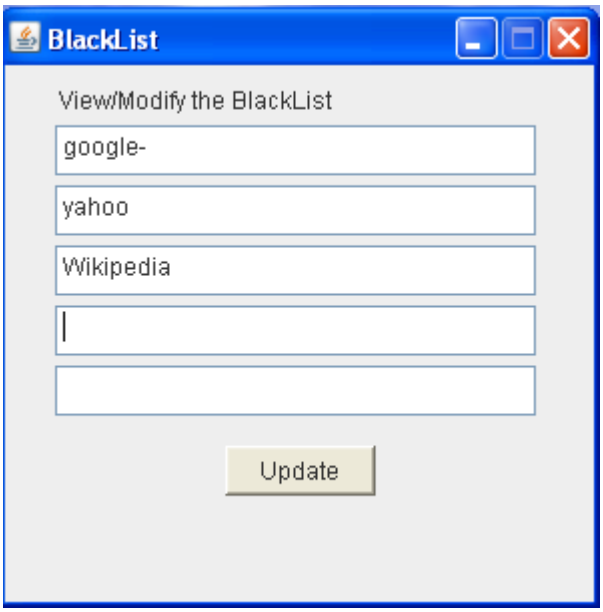

Figure 9: BlackList to update the header of the applications and website.

We can add the websites or the application name that the students are not supposed to open to the black list. Once we have given the list we can update it. When we click on the Update command button we call the method named createInsertQuery and hide the current GUI.

```
public void actionPerformed(ActionEvent ae1){
        Object ob=ae1.getSource();
                try {
                if(ob==add){
                createInsertQuery();
                this.setVisible(false);}
            }
catch (Exception ex) {}
    }
```

5.2.4. White List GUI

In our tool we also have the white list command button to update the white list. Once we click on the white list command button the small frame will be invoked as shown in the Figure.

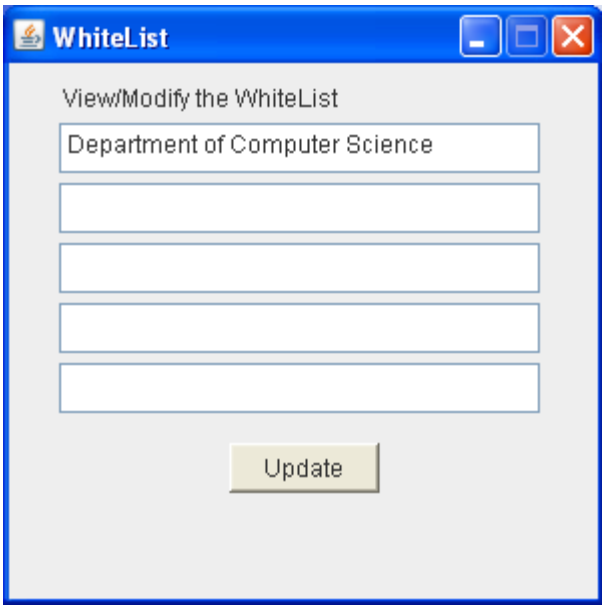

Figure 10: White List to update the header of the applications and website.

We can give the website or application names on the list which the students are allowed to open. Once we have given the list then we can update it. When we click on the Update command button we call the method named createInsertQuery and hide the current GUI in the same manner as the black list command button.

5.2.5. Log File

The log file will show every activity performed by each student. The professor will have the option to view the log files during class time while running the tool or after the class. If the professor wants to view the log file during the session he can do so by clicking View Log Files and the following code will be executed.

```
public  void loadLogList(){
      try{
                File folder = new File("c:/students_log");
        File[] listOfFiles = folder.listFiles();
        clientLogList.removeAll();
        logList.removeAll();
        for (int i = 0; i < listOfFiles.length; i++) {
          if (listOfFiles[i].isFile())
           { String s=listOfFiles[i].getName();
            clientLogList.add(s);}
          else if (listOfFiles[i].isDirectory())
            {System.out.println("Directory " + listOfFiles[i].getName());}  }  }
                catch(Exception e)
                {System.out.println ("Error getting file name");} }
```

A small window will be opened which displays all the client (student) machines that are connected to the network, as shown in figure 11.

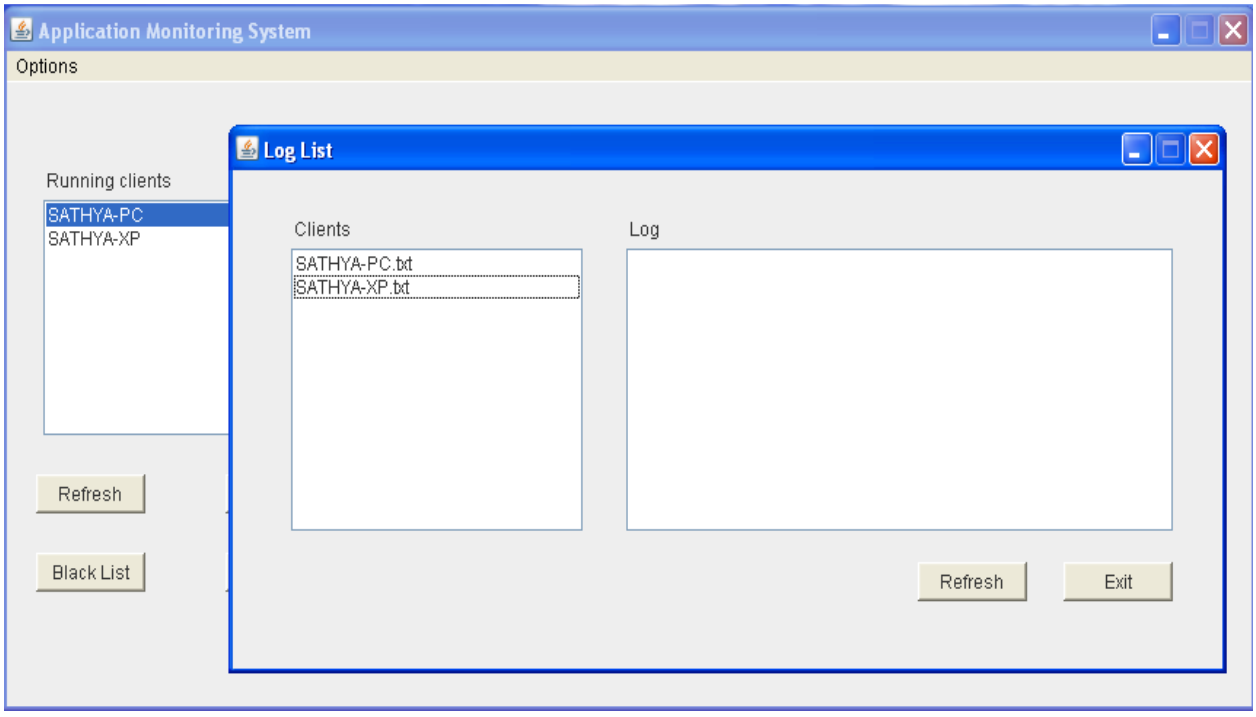

Figure 11: Log List during the session once clicked on the View log File.

If the professor clicks on a particular machine then the following code will be called to display the activities performed by that student, as shown in figure 12.

```
public void actionPerformed(ActionEvent ae2){
        Object ob2=ae2.getSource();
        logList.removeAll();
        if(ob2==clientLogList){
        try{File f1=new File("c:\\students_log\\"+clientLogList.getSelectedItem());
                FileReader fr=new FileReader(f1);
                BufferedReader br=new BufferedReader(fr);
                String s=null;
        do      {       s=br.readLine();
                logList.add(s);  }
        while(s!=null);}
    catch(Exception e){} } }
```

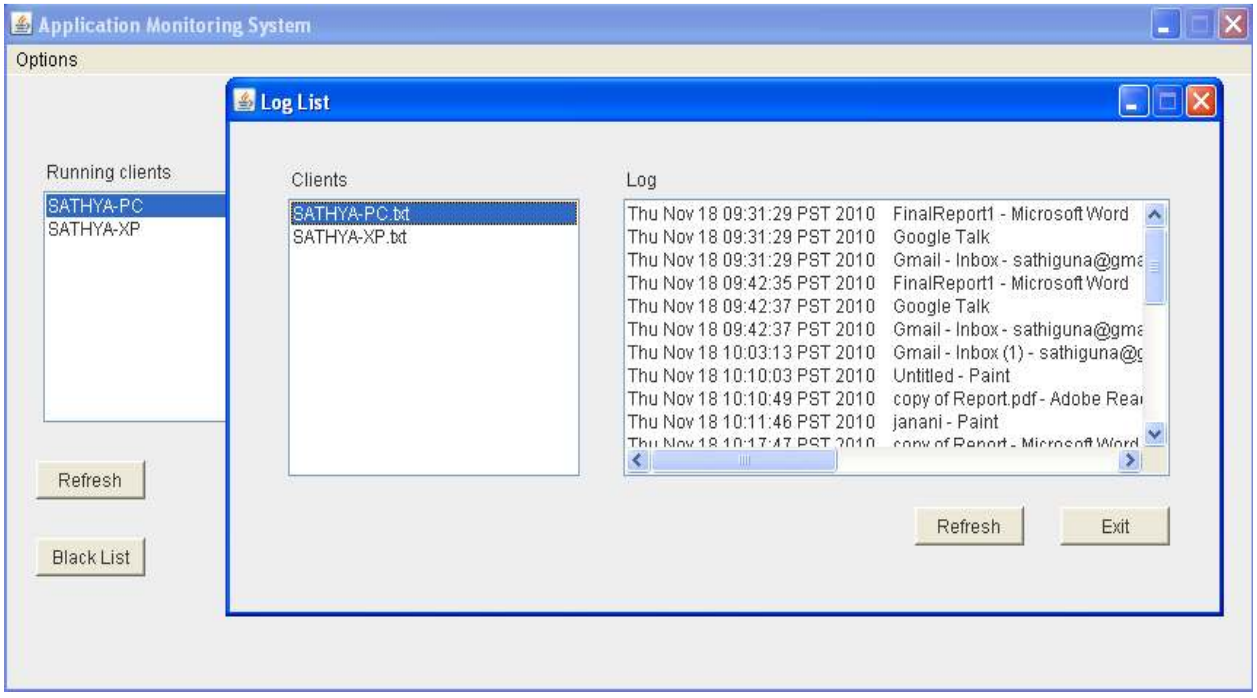

Figure 12: Professor selects particular client machine.

When a professor is looking into the log file he has an option to refresh and update the tool. The following code will be executed in order to update the tool.

```
else if(ob2==refreshLog)
 { this.loadLogList(); }
```

To exit the log file the professor clicks on the Exit command button, at which point the following code will be executed and the log view GUI will be hidden.

```
else if(ob2==exitLog)
  { this.setVisible(false); }
```

If the professor wants to see a particular student's log file after the session has ended, he can still do so. While the tool is running it will create a folder called students_log in C drive. This folder will have a separate text file for each client, with their host name, which is shown in the Running Client when the tool is running.

```
iClient=(IClient)Naming.lookup("rmi://"+clientList.get(m)+"/Client");
f=new File("C:\\students_log\\"+clientList.get(m).toString()+".txt");
```

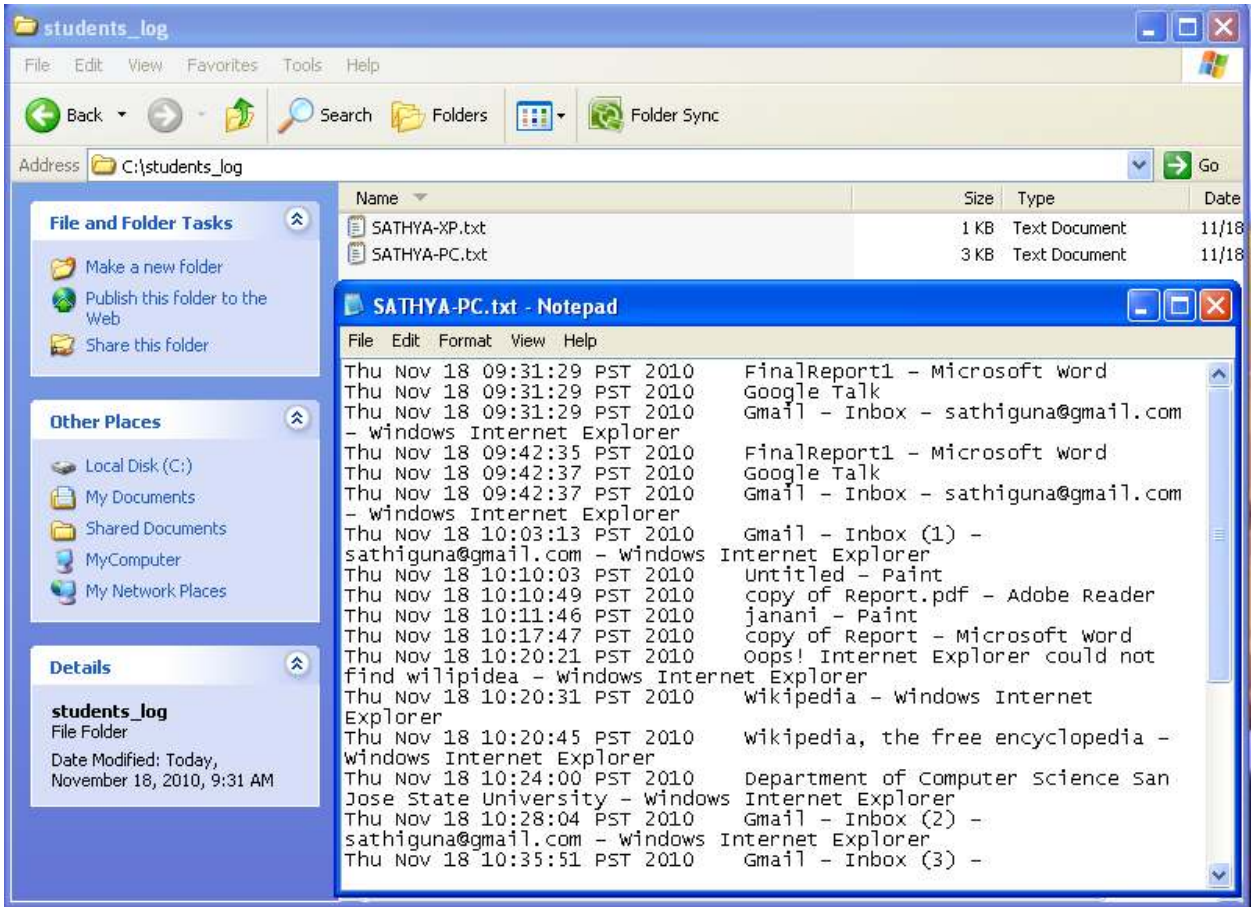

Figure 13: Log list after the session.

Each text file will record all the activities performed by a particular client (student) with the date and time.

```
fout=new FileWriter(f,true);
fout.write(new Date().toString()+" "+clientApplicationList.get(i).toString()+"\r\n");
```

5.2.8. Database

5.2.8.1. Database Connectivity

For our tool we are using Microsoft Access for the database where our black list and white list will be saved. The following getconnection() method is used for getting the database connectivity.

```
public void getCon() throws SQLException
{
   try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection("jdbc:odbc:list");
    st=con.createStatement();
      }
        catch (Exception ex)
        {System.out.println ("Connection Exception: " +ex);}
}
```

5.2.8.2. Commit the Database

We use comitIT(), which is commit method to do the commit operation in the database. After every query we have to call commit method to ensure all changes have been done.

```
public void comitIt() throws Exception
        { con.commit(); }
```

5.2.8.3. Terminate Database Connectivity

In order to close the connectivity we use the following method closeIT().

```
public void closeIt() throws Exception
   {     con.close();   }
```

5.2.9. Pop-up for Black Listed Website

If a professor has instructed students not to open certain websites, and if they open those websites, then the professor should receive a notification with the student host name and website address. By default we start our tool with the Blacklist radio button selected.

Once the client (student) machines are connected to our tool, it will check for the header's name, which is in the black list in the each client machine and is displayed in Running Clients. If it matches with the list, then it will open the pop-up. At the same time, if three client (student) machines open black listed websites, then the pop-up will be shown for all three machines.

If Blacklist is selected then the following code is used for getting the connection with the database and fetch method is called with the select query.

```
public static void getDB() throws Exception
   {
        DBWebList dbcon=new DBWebList();
        dbcon.getCon();
        dbweb=dbcon.fetch("select * from blacklisted");
   }
```

The following code is used to show the pop-up whenever a client (student) opens a black listed website.

```
if(flag==0)
        {
if((clientApplicationList.get(i).toString().toLowerCase().contains(bList[0].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[1].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[2].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[3].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[4].toLowerCase())))
                {
           msg=clientList.get(m) +" has opened "+ clientApplicationList.get(i).toString()+"!!";
           showAlert(msg);
                }
        }
```

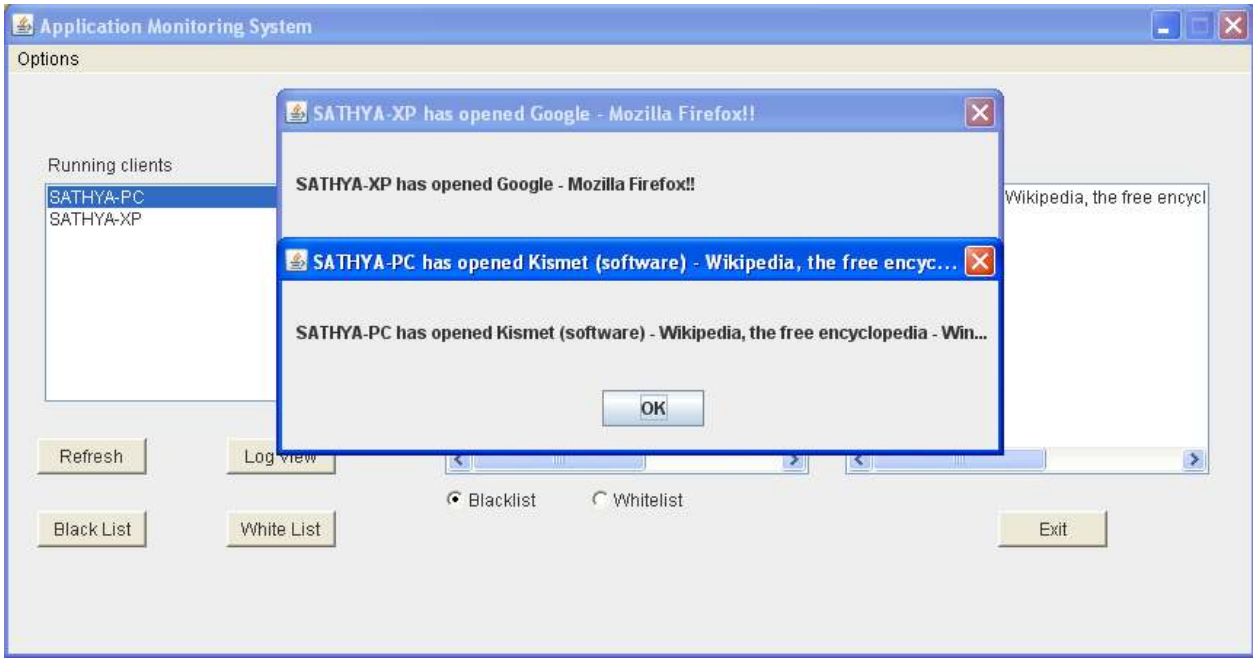

Figure 14: Screenshot for the BlackList Website pop-ups

5.2.10. Pop-up for Whitelisted Websites

If a professor has instructed students that they can open certain websites and if they open websites other than those permitted, the professor will receive a notification containing the student host name and website address. Once the client (student) machines are connected to our tool, it will check for the header name, which is in the white list in the each client machine. This is displayed in Running Clients. If it doesn't match with the white list then, it display the pop-up. Even if three students access websites other than the white list, the professor will get three pop-ups with the student host names and website addresses.

When the white list is selected the following code is used for establishing connection with the database, and fetch method is called with select query.

```
public static void getDB() throws Exception
  {
       DBWebList dbcon3=new DBWebList();
       dbcon3.getCon();
       dbweb=dbcon3.fetch("select * from whitelisted");
       dbcon3.closeIt();
  }
```

The following code is used to show a pop-up whenever a client (student) opens a website other than those on the white list.

```
if((clientApplicationList.get(i).toString().toLowerCase().contains(bList[0].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[1].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[2].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[3].toLowerCase()))||
(clientApplicationList.get(i).toString().toLowerCase().contains(bList[4].toLowerCase())))
                        {}
else
{
msg=clientList.get(m) +" has opened "+ clientApplicationList.get(i).toString()+"!!";
showAlert(msg);
}
```

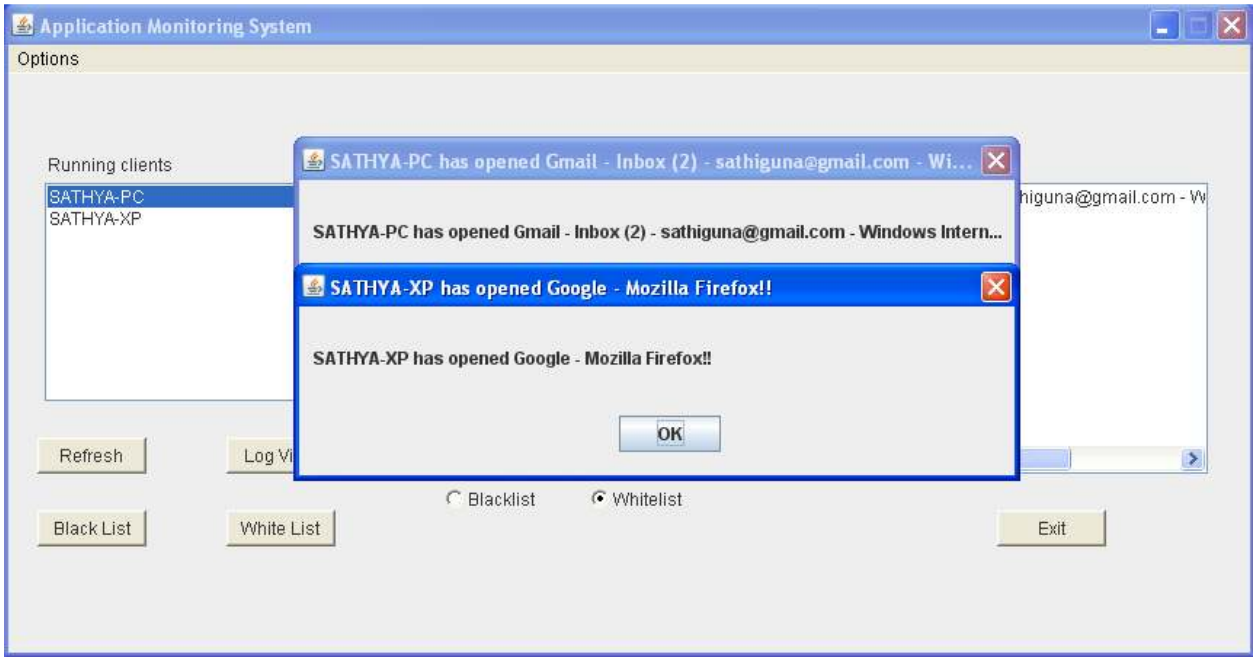

Figure 15: Screenshot for the Whitelisted Website Pop-ups

5.2.11. Exit

When a professor is done with the tool, then he clicks on the Exit command button to close it. The following code is used for the Exit command button.

```
if(ob==commandExit || ob==exit){
    try
        { Server.fout.close(); }
    catch(Exception e){ }
System.exit(0); }
```

6. Possible Attacks

There are some ways by which students can still cheat the tool. Possible attacks are given below.

1. Because students have to connect to the same network where the professor will give the SSID, a student could disconnect from the SSID given by the professor and connect to a different SSID, look for the answer, and then connect to the same SSID that was given by the professor.

For example, suppose the professor gave the SSID "TEST" to the students. Students connect to the "TEST" SSID and run the client-side program. The professor would be able to see the student's host name in the tool. Students can then disconnect from the "TEST" SSID and connect to a different SSID.

2. The tool is getting the applications or websites which are running from the task manager. If the student changes the name of the application, then the task manager will also have the same name that was given by the student.

For example, a student has changed the name of "Internet Explorer" to some other name, such as "FAKE," then the task manager will also have the same name.

3. If a student has two wireless cards in the laptop, then he can connect one wireless card to the SSID given by the professor and another one to any other network, and find the answers.

Suppose the professor gave the SSID of "TEST" to the students. If Student8 has two wireless cards in his laptop, he can connect to "TEST" and run the client-side program. The professor continues to see the Student8 host name on the tool. By using another wireless card, the student can then connect to a different network and find the answers.

7. Solution

7.1. Auto Refresh

One of the solutions for the attack when a student leaves the SSID given by the professor is to auto refresh the client (student) machines. Whenever students connect to the SSID given by the professor, the professor will be notified. For example, hostname has logged in. If any client goes off, the professor will be notified. For example, hostname has logged out.

We call getItem method of clientList to get the currently connected client names. The clients, that are already connected and shown in Running Clients will be placed in an array list. Once we refresh, we put them in a separate array list. We compare both the array lists to check for new client machines or if any client machine is missing. If there is a new client machine, then there will be a pop-up showing "logged in." If any client is missing then there will be a pop-up showing "logged out." The following code is used for client auto refresh.

```
try{
 clientArrayList.removeAll(clientArrayList);
 clientArrayList=server.getClients();
  for(int j=0;j<listContent.size();j++){
        if(!(clientArrayList.contains(listContent.get(j)))){
        String msg111=listContent.get(j)+" has logged Out";
        PopDialog pd=new PopDialog(fr1,false,msg111);} }
  for(int j=0;j<clientArrayList.size();j++){
        if(!(listContent.contains(clientArrayList.get(j)))){
        String msg112=clientArrayList.get(j)+" has logged In";
        PopDialog pd=new PopDialog(fr1,false,msg112);}}
  clientList.removeAll();
 catch(Exception e){System.out.println(e);
            e.printStackTrace(); } }
```

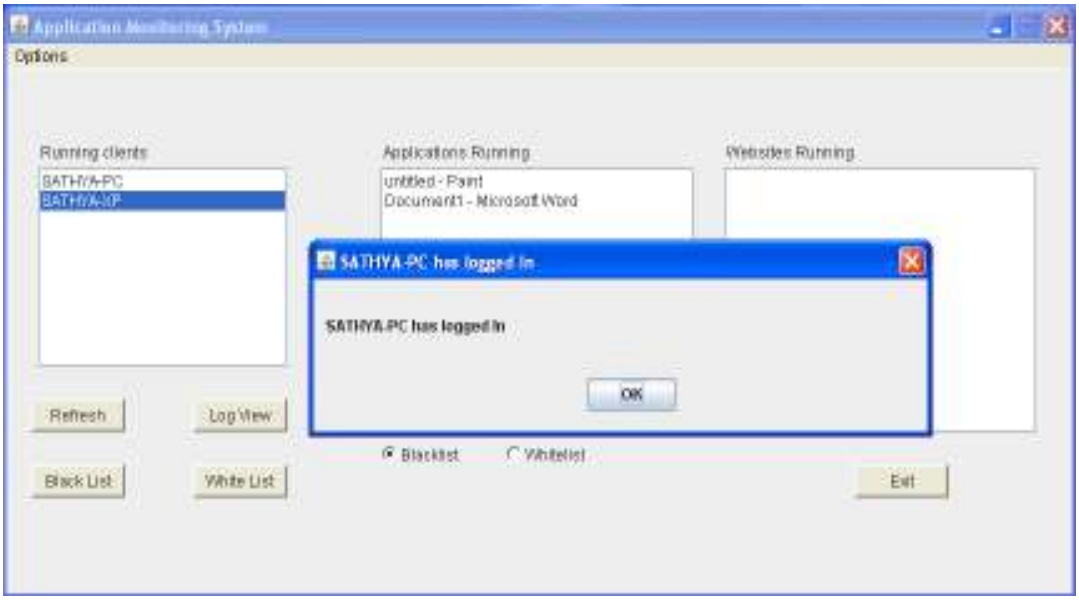

Figure 16: Pop-up to show the Student Logged In

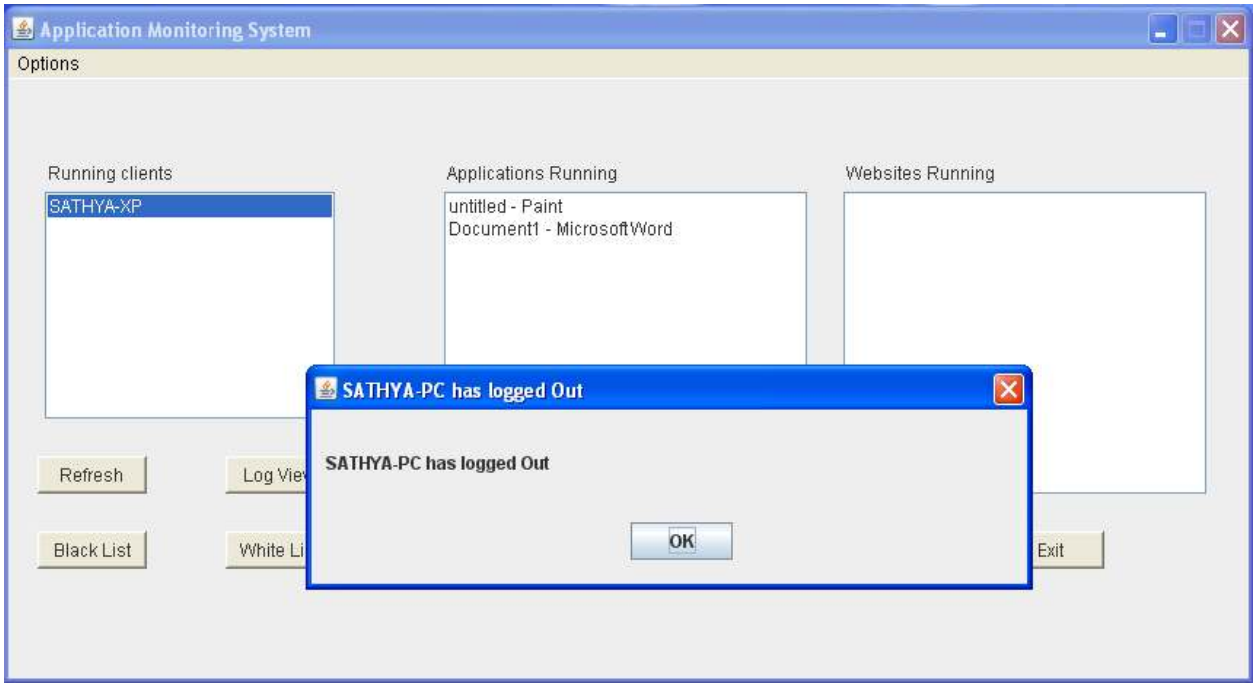

Figure 17: Pop-up to show the Student Logged Out.

While doing auto refresh, we have to change the NegativeCacheTime to one second. The server will be holding the NegativeCacheTime for some minutes by default [10]. We need to refresh very frequently, so we add the NegativeCacheTime as one second in the system registry.

Steps to change or add the NegativeCacheTime is as follows:

1.Goto start menu, click on run, and type regedit. [11]

2.Once the Registory Editory is opened then goto HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSetServices\Dnscache\Parameters [12].

3.If the NegativeCacheTime is not there, then build a new DWORD as NegativeCacheTime.

4. Then change the value to 1 sec. [13]

The above steps are used to increase system speed when the client machine logs in or logs off.

8. Testing

We did the testing in one class to find out the efficiency of the tool. Testing of the major features is described below.

Viewing List of Student Computers in the Command Prompt

The professor provides the SSID and the other requirements to the students to connect to the network and gives the "Net View" command in the command prompt. The list of students who are connected to the SSID is shown.

Executing the Tool

Students start the client-side program and the professor starts the server-side program. The tool is opened in the professor's machine with the clients running. A pop-up will appear showing the student host names of any who started the client late.

Updating the Black List and White List

The professor clicks on the black list, and once the pop-up is shown, he updates the list. He updates the white list in the same manner.

Select the List

Pop-ups will be shown on the professors' screen based upon which list is selected.

View the Log File

The professor clicks on the log file and on the particular student's host name. This displays all the activities performed by the student.

The below table shows the activities performed by students and the corresponding system result.

No.	Students Activities	Online Monitoring Tool Response	Pass/Fail
1.	Students connect to the SSID which was given by the professor and run the client side program		Pass
2.	Student (Hostname: Student5) came late to the class, connects to the SSID and runs the client-side program.	Should show the pop-up that, "Student5 logged in. "	Pass
3.	Student9 opens a black listed website	There should be a pop-up that "Student9 has opened the website name."	Pass
4.	Student2 disconnected from the SSID which was given by the professor and connected to a different SSID.	There should be a pop-up that, "Student2 Logged Out."	Pass

5.	Student7 opened the website other than from the white listed.	There should be a pop-up that, "Student7 opened the website name'.	Pass
6.	Student1 Closed the client-side program.	There should be a pop-up that, "Student1 Logged Out."	Pass

Table 1: Activities performed by students.

The Table below shows the activities performed by the professor and the corresponding system response.

No	Professor Activities	Online Monitoring Tool Response	Pass/Fail
1.	Professor runs the server-side program.	Tool should open with the Blacklist option selected. It should have the list of the Clients Running, which was started before starting the server program.	Pass
2.	Professor clicks on the Blacklist button.	Small window should open to enter the website header name.	Pass
3.	Professor clicks update on the small window.	The list should be saved in the tool.	Pass
4.	Professor double clicks on the Student5 on the Clients Running.	In the Applications Running it should show the applications and websites opened by the student.	Pass

5.	Professor clicks on the View Log button.	Window must have opened with the list of students connected.	Pass
6.	Professor clicks on a particular student file.	It will show all the details from the student.	Pass
7.	Professor clicks on the Whitelist button.	Small Window show open to enter the header name of the website address.	Pass
8.	Professor clicks on the update button from the list.	The list should be saved in the tool.	Pass
9.	Professor clicks on the refresh button.	Tool will refresh and give you the updated Clients Running.	Pass
10.	Once the class is over the professor can see the Log File from the C:\Student_Log.	There should be separate NotePad with each student name.	Pass
11.	Professor clicks on the particular student name.	It should display all the activities performed by the student during the test.	Pass

Table 2: Activities performed by professor.

9. Conclusions and Future Work

The goal of this project is to catch students when they try to cheat during computer testing. So by using the Blacklist and Whitelist buttons, professor can determine for students are cheating during the exam. The professor can also see what a particular student is doing during the entire class by using the log file during or after the session.

Our tool only works for the Microsoft Windows Operating System. We should develop a tool that will work for all operating systems, such as Linux and Mac OS [19].

There should be a tool that can retrieve information from highly protected client (student) machines.

We should be able to get the information without asking the students to execute the programs. Basically, students shouldn't have any knowledge that the professor is monitoring their online activities, or how that monitoring is being carried out.

The tool should work if the student is connected to the network without requiring the student to connect to a specific gateway.

10. References

[1] Ulf Lamping, Richard Sharpe, Ed Warnicke, (2004-2008). *Wireshark User's Guide 31757 for Wireshark 1.2.*

[2] Gerald Combs, Lead Developer, Wireshark Director, CACE technologies, (2009). *Introduction to Wireshark*
http://media-2.cacetech.com/video/wireshark/introduction-to-wireshark/

[3] R. Philip, (2007). *Securing wireless networks from ARP cache poisoning.*
http://www.cs.sjsu.edu/faculty/stamp/students/Roney298report.pdf

[4] Larry L Peterson, and Bruce S. Davie, (2007). *Computer Networks A System Approach.*USA:Morgan Kaufmann Publication.

[5] Kismet Documentation.
http://www.kismetwireless.net/documentation.shtml

[6] Client Server Architecture
http://www.utdallas.edu/~chung/SA/2client.pdf

[7] Windy 31
http://www.synetusa.com/

[8] Windows Product Documentation: Net View

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/net_view.mspx?mfr=true

[9] Remote Method Invocation:
http://java.sun.com/developer/onlineTraining/rmi/RMI.html

[10] Clear Windows DNS cache
http://www.tech-recipes.com/rx/233/clear_windows_dns_cache/

[11] Beginners Guides: The Registry: Backups, Repairs, and Protection
http://www.pcstats.com/articleview.cfm?articleID=263

[12] How to make your windows run super fast
http://www.ihackintosh.com/2009/03/how-to-make-your-windows-run-superfast/]

[13] How to flush DNS
http://www.tech-faq.com/how-to-flush-dns.html

[14] Sharing files in the network
http://www.howtogeek.com/howto/windows-7/share-files-and-printers-between-windows-7-and-xp/

[15] Focus on Security.An Overview of Non-Commercial Software for Network Administration.
http://uccsc2009.ucdavis.edu/preso/NomuraUCCSC09.ppt

[16] Portable Wireless USB Router : DigInfo
http://www.youtube.com/watch?v=JQ4qZ9Dgq3I

[17] Mark Stamp, (2009). *Information Security Principles and Practice, exam questions and answer.*

[18] Larry L Peterson, and Bruce S. Davie, (2007). *Computer Networks A System Approach.*USA:Morgan Kaufmann Publication.

[19] Mac OS
http://www.apple.com/macosx/

[20] Service Set Identifier.
http://compnetworking.about.com/cs/wireless/g/bldef_ssid.htm