PROJECT REPORT
ON Network Sniffer Build

CHIKKALA SOUJANYA

CONTENTS

OBJECTIVE

Network packet sniffer or simply packet sniffer is a packet analyzer software that monitors all network traffic. The proposed project is implemented in Java programming language, and using this application admin of the system can capture network packet and analyze data received/sent from/to the network.

Developed as a desktop application, packet sniffer facilitates web-based monitoring of network packets which are traveling over the system network. The primary data captured by this software is the packets source and destination addresses.

In this article, the project has been briefly discussed explaining its scope, features, and system specifications.

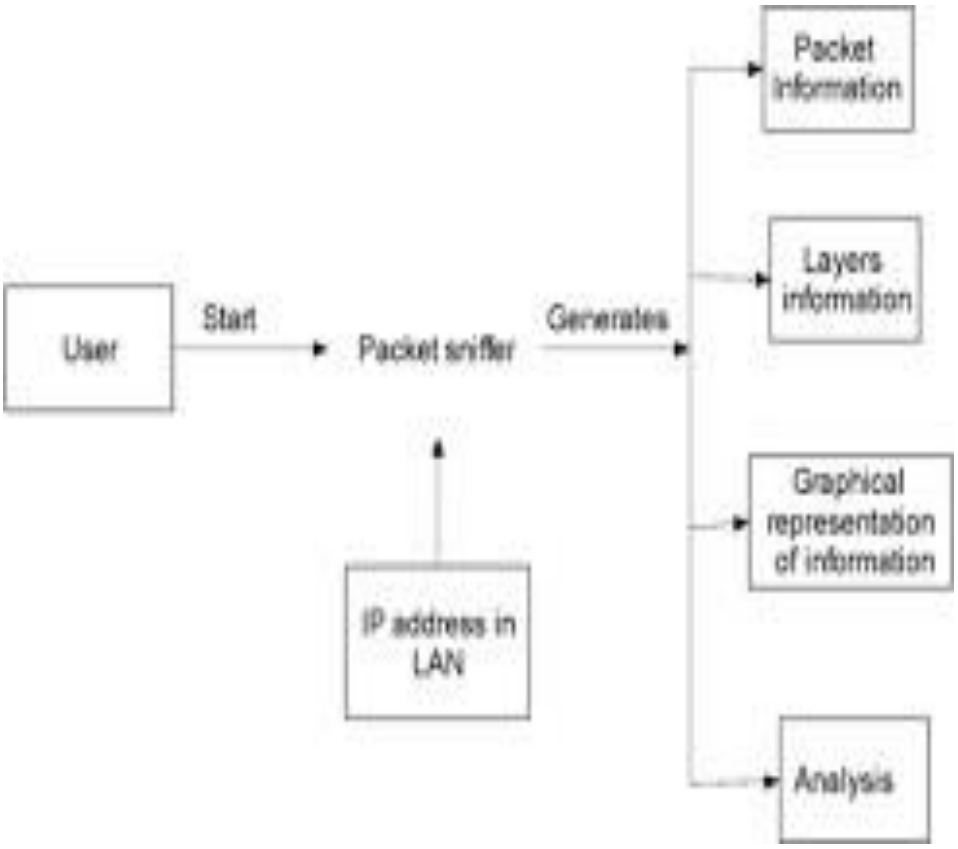
PROBLEM STATEMENT

Network packet sniffer is simply a web-based application that monitors all traffic over a network. Unlike other standard network hosts that only track traffic sent particularly to them, this software captures each packet, eventually decoding and analyzing its data as the data streams flow across the system network.

This project, developed in Java, shows mainly two things:

1. how real-time network connection behavior can be modeled as chromosomes
2. how the parameters in genetic algorithm can be defined in this respect.

DATAFLOW DIAGRAM



SAMPLE CODING

```
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.net.*;
public class AboutDialog extends JDialog
{
```

```
private JScrollPane jsp;
private JEditorPane helpfile;
public AboutDialog(JFrame owner)
{
    super(owner, "About Schnufflen");
    URL fileurl=null;
    File file=null;
    helpfile = new JEditorPane();
    helpfile.setEditable(false);
    helpfile.setContentType("text/html");
    try
    {
        fileurl = MainGui.class.getResource("README.htm");
        helpfile.setPage(fileurl);
    }
    catch(IOException ex)
    {
        ex.printStackTrace();
    }
    jsp = new JScrollPane();
    jsp.getViewport().add(helpfile, BorderLayout.CENTER);
    setSize(600, 300);
    getContentPane().add(jsp);
```

```
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setVisible(true);
    }
}


public class listdata
{
public String header;
public String data;

public String toString()
{
        return header;
}
}

import java.io.File;
public class LogFilter extends javax.swing.filechooser.FileFilter
{
    private String xtnsn;
    public LogFilter(String str)
    {
```

```
        if(str!=null)
        {
            xtnsn=str;
        }
        else
        {
            xtnsn=null;
        }
    }
    public boolean accept(File f)
   {
      if(f.isDirectory())
      {
        return true;
      }
      if(getExtension(f).equalsIgnoreCase(xtnsn))
      {
        return true;
      }
      return false;
    }

  public String getDescription()
```

```
  {
     return xtnsn;
  }
private String getExtension(File f)
  {
     String s = f.getName();
     int i = s.lastIndexOf('.');
     if (i > 0 &&  i < s.length() - 1)
     {
        return s.substring(i).toLowerCase();
     }
     return "";
  }
protected void finalize()
  {
    xtnsn=null;
    }
}

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
import javax.swing.event.*;

import jpcap.*;

public class MainGui extends JFrame implements MouseListener,
ListSelectionListener, ActionListener

{

   private JPopupMenu popup;

    public JPopupMenu getpopup(){ return popup; }

    public void setpopup(JPopupMenu jpm){ popup = jpm; }

     private snifferthread dt;

    public snifferthread getsniffer(){return dt;}

    public void setsniffer(snifferthread pdt){dt=pdt;}


    private DefaultListModel model;

    public DefaultListModel getModel(){ return model; }

    public void setModel(DefaultListModel dlm){ model = dlm; }


    private int type;

    public int getType(){ return type; }

    public void setType(int tprm){ type = tprm; }


    private JScrollPane jsp1;

    public JScrollPane getScrollPane1(){ return jsp1; }

    public void setScrollPane1(JScrollPane pjsp){ jsp1=pjsp; }
```

```
private JScrollPane jsp2;
public JScrollPane getScrollPane2(){ return jsp2; }
public void setScrollPane2(JScrollPane pjsp){ jsp2=pjsp; }

private JTextArea jt1;
public JTextArea getTextArea1(){ return jt1; }
public void setTextArea1(JTextArea pjt){ jt1=pjt; }

private JList jt2;
public JList getTextArea2(){ return jt2; }
public void setTextArea2(JList pjt){ jt2=pjt; }

private JComboBox nic;
public JComboBox getComboBox(){ return nic; }
public void setComboBox(JComboBox pjc){ nic=pjc; }

private DefaultListModel nicmdl;
public DefaultListModel getNicmdl(){ return nicmdl; }

private UIManager.LookAndFeelInfo[] landf;
public UIManager.LookAndFeelInfo[] getLandF(){ return landf; }
```

```
public void setLandF(UIManager.LookAndFeelInfo[] plandf){ landf=plandf; }

private mainmenuhandler mmh;

public mainmenuhandler getMenuHandler(){ return mmh; }

public void setMenuHandler(mainmenuhandler pmmh){ mmh=pmmh; }

public MainGui()

{

    java.net.URL imageURL=null;

    ImageIcon img=null;

    JMenuItem menuItem=null;

    popup=null;

    dt = null;

    jt1=null;

    jt2=null;

    jsp1=null;

    jsp2=null;

    landf=null;

    mmh=null;

    type=-1;

    NetworkInterface[] devices = JpcapCaptor.getDeviceList();

    String[] interfaces = new String[devices.length+1];

    int i=0;

    try
```

```
        {
            imageURL                                                    =
MainGui.class.getResource("images/sourcecon.jpg");
            img = new ImageIcon(imageURL);

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClas
sName());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        setSize(300, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        interfaces[i++]=new String("Please select a network interface.");
        for(; i < devices.length+1; i++)
        {
            interfaces[i]=new String(i-1 + ": " + devices[i-1].name + "(" +
devices[i-1].description + ")");
        }
        nic = new JComboBox(interfaces);
        nic.addActionListener(this);
        nic.setBorder(BorderFactory.createLoweredBevelBorder());
        nic.setMinimumSize(new Dimension(300,30));
```

```
        getContentPane().add(nic, BorderLayout.PAGE_START);
        jt1 = new JTextArea();
        jt1.addMouseListener(this);
        jt1.setLineWrap(true);
        jt1.setMinimumSize(new Dimension(150, 150));
        jsp1=new JScrollPane(jt1);
        model = new DefaultListModel();
        jt2 = new JList(model);
        jt2.addMouseListener(this);
        jt2.setMinimumSize(new Dimension(150, 50));

jt2.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        jt2.addListSelectionListener(this);
        jsp2=new JScrollPane(jt2);
        JSplitPane sp = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT, jsp2, jsp1);
        getContentPane().add(sp, BorderLayout.CENTER);
        setIconImage(img.getImage());
        sp.setDividerLocation(100);
        mmh = new mainmenuhandler(this);
    }

    public void mouseClicked(MouseEvent e)
    {}public void mouseEntered(MouseEvent e)
```

```
 {}
 public void mouseExited(MouseEvent e)
 {}
 public void mousePressed(MouseEvent e)
 {}
  public void mouseReleased(MouseEvent e)
 {
     int i = e.getButton();
     if(i==MouseEvent.BUTTON3)
     {
         popup.show(e.getComponent(),e.getX(), e.getY());
     }
 }
public void actionPerformed(ActionEvent e)
 {
     if(e.getActionCommand().equals("comboBoxChanged"))
     {
         type = nic.getSelectedIndex()-1;
     }
     jt2.requestFocusInWindow();
 }
 public void valueChanged(ListSelectionEvent e)
 {
```

```
        listdata tmp;
        if(e.getValueIsAdjusting()== false)
        {
           if(jt2.getSelectedIndex()!=-1)
           {
              tmp=(listdata)jt2.getSelectedValue();
              jt1.setText(tmp.data);
              jt1.setCaretPosition(0);
           }
        }
    }
}

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class mainmenuhandler implements ActionListener
{
    private UIManager.LookAndFeelInfo[] landf;
    private MainGui frame;
    private JMenuBar menubar;
    private JMenu landfmenu;
    private JMenu sniffermenu;
    private JMenu aboutmenu;
    private JPopupMenu popup;
  public mainmenuhandler(MainGui frm)
    {
```

```
        JMenuItem menuItem=null;
        landf = UIManager.getInstalledLookAndFeels();
        frame = frm;
        menubar = new JMenuBar();
        popup = new JPopupMenu();
        aboutmenu = new JMenu("About");
        aboutmenu.setMnemonic(KeyEvent.VK_A);
        sniffermenu = new JMenu("Packet Sniffing");
        sniffermenu.setMnemonic(KeyEvent.VK_P);
        landfmenu = new JMenu("Look And Feel");
        landfmenu.setMnemonic(KeyEvent.VK_L);
        menuItem = new JMenuItem("About Schnufflen");
        menuItem.addActionListener(this);
        aboutmenu.add(menuItem);
        landf = UIManager.getInstalledLookAndFeels();
        for(int j = 0; j < landf.length; j++)
        {
           menuItem = new
JMenuItem(getclassname(landf[j].getClassName()));
           menuItem.addActionListener(this);
           landfmenu.add(menuItem);
        }
        menuItem = new JMenuItem("Start Logging Hex");
        menuItem.addActionListener(this);
        popup.add(menuItem);
        menuItem = new JMenuItem("Start Logging Hex");
        menuItem.addActionListener(this);
        sniffermenu.add(menuItem);
        menuItem = new JMenuItem("Start Logging Chars");
        menuItem.addActionListener(this);
        popup.add(menuItem);
        menuItem = new JMenuItem("Start Logging Chars");
        menuItem.addActionListener(this);
        sniffermenu.add(menuItem);
        menuItem = new JMenuItem("Stop Logging");
```

```
        menuItem.addActionListener(this);
        popup.add(menuItem);
        menuItem = new JMenuItem("Stop Logging");
        menuItem.addActionListener(this);
        sniffermenu.add(menuItem);
        menuItem = new JMenuItem("Save Log");
        menuItem.addActionListener(this);
        popup.add(menuItem);
        menuItem = new JMenuItem("Save Log");
        menuItem.addActionListener(this);
        sniffermenu.add(menuItem);
        menuItem = new JMenuItem("Exit");
        menuItem.addActionListener(this);
        popup.add(menuItem);
        menuItem = new JMenuItem("Exit");
        menuItem.addActionListener(this);
        sniffermenu.add(menuItem);
        menubar.add(sniffermenu);
        menubar.add(landfmenu);
        menubar.add(aboutmenu);
        frame.setpopup(popup);
        frame.setJMenuBar(menubar);
    }
    String getclassname(String originalname)
    {
        return
originalname.substring(originalname.lastIndexOf(".")+1);
    }
    public void actionPerformed(ActionEvent e)
    {
        FileOutputStream out=null;
        int sz=0;
        listdata lst=null;
        String dst=null;
        JFileChooser fs=null;
```

```
        File tmpfl=null;
        File selectedFile=null;
        String strtmp=null;
        String savdir=null;
        AboutDialog ad = null;
        snifferthread tmpthrd = null;
        try
        {
           for(int i = 0; i < landf.length; i++)
           {

if(getclassname(landf[i].getClassName()).equals(e.getActionCo
mmand()))
              {

UIManager.setLookAndFeel(landf[i].getClassName());
                 SwingUtilities.updateComponentTreeUI(frame);
                 break;
              }
           }
        }
        catch(Exception ex)
        {
           ex.printStackTrace();
        }
        if(e.getActionCommand().equals("comboBoxChanged"))
        {

frame.setType(frame.getComboBox().getSelectedIndex()-1);
        }
        else if(e.getActionCommand().equals("Start Logging
Hex"))
        {
           if(frame.getType()==-1)
           {
```

```
            JOptionPane.showMessageDialog(frame, "Please
select a network interface.", "No Network Interface Selected",
JOptionPane.ERROR_MESSAGE);
            return;
          }
          frame.getModel().clear();
          frame.getTextArea2().updateUI();
          frame.getTextArea1().setText("");
          if(frame.getsniffer()!=null)
          {
            frame.getsniffer().stopthread();
            frame.setsniffer(null);
          }
          tmpthrd = new snifferthread(frame.getType(),
frame.getModel(), true);
          frame.setsniffer(tmpthrd);
          tmpthrd.start();
        }
        else if(e.getActionCommand().equals("Start Logging
Chars"))
        {
          if(frame.getType()==-1)
          {
            JOptionPane.showMessageDialog(frame, "Please
select a network interface.", "No Network Interface Selected",
JOptionPane.ERROR_MESSAGE);
            return;
          }
          frame.getModel().clear();
          frame.getTextArea2().updateUI();
          frame.getTextArea1().setText("");
          if(frame.getsniffer()!=null)
          {
            frame.getsniffer().stopthread();
            frame.setsniffer(null);
```

```
            }
            tmpthrd = new snifferthread(frame.getType(),
frame.getModel(), false);
            frame.setsniffer(tmpthrd);
            tmpthrd.start();
        }
        else if(e.getActionCommand().equals("Stop Logging"))
        {
            if(frame.getsniffer()!=null)
            {
                frame.getsniffer().stopthread();
                frame.setsniffer(null);
            }
        }
        else if(e.getActionCommand().equals("Save Log"))
        {
            if(frame.getsniffer()!=null)
            {
                frame.getsniffer().stopthread();
                frame.setsniffer(null);
            }
            sz=frame.getModel().size();
            fs = new JFileChooser();
            fs.addChoosableFileFilter(new LogFilter(".log"));
            fs.setAlignmentX(frame.getAlignmentX());
            fs.setAlignmentY(frame.getAlignmentY());
            fs.setSelectedFile(null);
            if(fs.showSaveDialog(frame) ==
JFileChooser.APPROVE_OPTION)
            {
                savdir=fs.getCurrentDirectory().toString();
                tmpfl=fs.getSelectedFile();
                if(tmpfl!=null)
                {
                    strtmp=tmpfl.toString();
```

```
                if(strtmp.indexOf('.')>0)
                {

strtmp=strtmp.substring(0,strtmp.lastIndexOf('.'));
                }
                selectedFile = new File(strtmp +
fs.getFileFilter().getDescription());
                try
                {
                    if(selectedFile!=null)
                    {
                      try
                      {
                         out = new FileOutputStream(selectedFile);
                         for(int i=0;i<sz;i++)
                         {
                            lst=(listdata)frame.getModel().get(i);
                            dst = lst.header + "\r\n" + lst.data +
"\r\n\r\n----------------------------------\r\n\r\n";
                            out.write(dst.getBytes(),0,dst.length());
                          }
                          out.close();
                          frame.getModel().removeAllElements();
                          frame.getTextArea1().setText("");
                       }
                       catch(Exception errr)
                       {
                          System.out.println(errr.toString());
                       }
                    }
                 }
                 catch(Exception err)
                 {
                    System.out.println(err.toString());
                 }
```

```
            }
        }
    }
    else if(e.getActionCommand().equals("Exit"))
    {
        System.exit(0);
    }
    else if(e.getActionCommand().equals("About
Schnufflen"))
    {
        ad = new AboutDialog(frame);
    }
  }
}
```

SCREEN SHOT: