

**LIGHTWEIGHT NETWORK MONITORING SYSTEM (NMS)  
USING SIMPLE NETWORK MANAGEMENT PROTOCOL  
(SNMP)**

**A**

**Project Report**

*submitted in partial fulfilment of the  
requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE**

**Specialization in**

**Oil and Gas Informatics**

**By:**

Name	Roll No	Branch
Abhishek Suman	R970216004	CSE OGI

*Under the guidance of*

**Dr. Kingshuk Srivastava**

**Assistant Professor (SG)**

**Department of Informatics**



**Department of Informatics**

**School of Computer Science**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**Bidholi, Via Prem Nagar, Dehradun, Uttarakhand**

**2019-20**



## CANDIDATES DECLARATION

I hereby certify that the project work entitled **LIGHTWEIGHT NETWORK MONITORING SYSTEM (NMS) USING SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Oil and Gas Informatics and submitted to the Department of Informatics at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **January, 2020 to April, 2020** under the supervision of **Dr. Kingshuk Srivastava, Assistant Professor(SG), Department of Informatics** .  
The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

**Name of Student(s)**      Abhishek Suman  
**Roll No.**                      R970216004

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 16 April 2020)

**(Dr. Kingshuk Srivastava)**  
Project Guide

**Dr. Thipendra Pal Singh**  
Head  
Department of Informatics  
School of Computer Science  
University of Petroleum and Energy Studies

## **ACKNOWLEDGEMENT**

I wish to express our deep gratitude to our guide **Dr. Kingshuk Srivastava**, for all advice, encouragement and constant support he has given me throughout our project work. This work would not have been possible without his support and valuable suggestions.

I sincerely thank to our Head of the Department, **Dr. Thipendra Pal Singh**, for his great support in doing my **Lightweight Network Monitoring System at SoCS**.

I am also grateful to **Dr. Manish Prateek, Dean SoCS**, UPES for giving me the necessary facilities to carry out my project work successfully.

I would like to thank all my **friends** for their help and constructive criticism during my project work. Finally, I have no words to express my sincere gratitude to my **parents** who have shown me this world and for every support they have given me.

<b>Name of Student(s)</b>	Abhishek Suman
<b>Roll No.</b>	R970216004

## **ABSTRACT**

In earlier days, to manage small network and its need simple network management protocol was present to implement basic network-management tool. The progress in OSI was slow for system management, which was later filled by SNMP. By today, Simple Network Management Protocol has become the most important network monitoring and managing asset. SNMP provides a utility to define and manage the network information and also for the information exchange. It provides agents and managers for network monitoring. An agent is set up into the managed devices and manager is set up into network monitoring system. Not only with the SNMP enabled devices but it also provides management of devices which are SNMP disabled. It uses proxies to achieve it. Proxy serve as an SNMP agent that collects and monitor information from more than one non-SNMP devices. Agent provides a management information base (MIB) which contains collection of objects categorised into groups. This project will provide a graphical user interface, which displays the system information. It can also be used to view the connection type and it will describe the port scanning method using socket programming.

**Keywords:** SNMP, OSI, TCP, UDP.

# TABLE OF CONTENTS

## Contents

1 Introduction	1
2 Background Study	2
3 Problem Statement	2
4 Objective	3
5 Methodology	3
6 Design	3
7 Implementation	4
7.1 Pseudo Code .....	4
7.1.1 Import necessary libraries .....	4
7.1.2 Monitor Definition .....	4
7.1.3 Monitor Thread .....	5
7.1.4 Run Monitor .....	5
7.1.5 Start Monitor .....	5
7.1.6 Stop Monitor .....	6
7.1.7 Network Monitoring Request .....	6
7.1.8 Network Monitoring Response .....	6
7.1.9 Test URL .....	6
7.1.10 Test IP .....	7
7.2 Output Screen .....	7
7.3 Result Analysis .....	8
8 Conclusion and Future Scope	10
A APPENDIX PROJECT CODE	12

# LIST OF FIGURES

## List of Figures

1	<i>Use Case Diagram of network monitoring system</i>	3
2	<i>Activity Diagram of network monitoring system</i>	4
3	<i>Importing Necessary Libraries</i>	4
4	<i>Monitor Definition</i>	5
5	<i>Monitor Thread</i>	5
6	<i>Run Monitor</i>	5
7	<i>Start Monitor</i>	5
8	<i>Stop Monitor</i>	6
9	<i>Network Monitoring Request</i>	6
10	<i>Network Monitoring Response</i>	6
11	<i>Test URL</i>	6
12	<i>Test IP</i>	7
13	<i>Monitoring of URL when connected</i>	7
14	<i>Monitoring of URL when network is not connected</i>	7
15	<i>Alert when disconnected</i>	8
16	<i>Monitoring of VPN</i>	8
17	<i>Alert when VPN is not present</i>	8
18	<i>Local Area Network</i>	9
19	<i>Device Availability</i>	9
20	<i>Service Availability</i>	9

# 1 Introduction

The appearance of PC's, workstations, LANs and servers changed the state of systems for eternity. Where once there were imbecilic terminals and a bunch of wise hosts, networks of clever frameworks grouped together and afterward contacted speak with each other. The market reacted with a cornucopia of new gadgets nearby and remote extensions, multiprotocol switches, appropriated centre points, and exchanging centres. The higher transmission capacity prerequisites of LAN-to-LAN organizing acquired elite media communications gear, for example, TI DSU/CSU units or casing transfer interfaces.

Clients started to purchase frameworks and hardware from a wide range of merchants. At the point when clients requested that the merchants supply the way to arrange, screen and test organize gear; every seller created a comfort item that conversed with its hardware utilizing a hand-made mystery language.

Each time another item was acquainted with the earth, another UI elbowed its way into the jam-packed Network Operations Center. Every UI came total with its own wording, baffling order phrases, and navigational rationale. Each apparatus estimated and checked by various principles. It appeared to be improbable that the circumstance could do anything besides get consistently increasingly disorganized and all the more befuddling. Yet, today we have a generally actualized convention for arrange the board the Simple Network Management Protocol or SNMP.

The SNMP model accept the presence of administrators and operators. A chief is a product module in an administration framework liable for overseeing part or the entirety of the design for the benefit of system the board applications and clients. A specialist is a product module in an oversight gadget liable for keeping up neighbourhood the executive's data to a trough by means of SNMP. An administration data trade can be started by the director or by the operator (trap). Inside the SNMP system, the executive's data is spoken to utilizing Abstract Syntax Notation (ASN.1). An administration data base comprises of an assortment of items sorted out into gatherings. Articles hold esteems that speak to oversight assets: a gathering is a unit of conformance.

There has been a quick and compelling co-activity in building up a typical language and a typical arrangement of estimations for organize hardware of numerous kinds. Gadgets running in advancement from repeaters to supercomputers contain the standard programming that they need so as to partake in arrange the board. System gadgets contain data about themselves. For instance, each gadget has been designed with some choice of parameters. A gadget has a present status that shows whether it is in sound running condition. Gadgets regularly keep inner insights that tally approaching and active traffic and different watched mistakes.

## 2 Background Study

The resources and papers referred are as follows:

1. Christina Fragouli, Athina Markopoulou (2005). **A Network Coding Approach to Overlay Network Monitoring**- *Stanford University*; Checking and determination of system conditions is a focal issue in systems administration. In that capacity, it has gotten a great deal of consideration in the Internet people group all in all and with regards to overlay organizes specifically. Autonomously, in organize coding have demonstrated that it is conceivable to expand arrange limit and better offer the accessible assets by permitting middle hubs to perform handling tasks, notwithstanding simply sending parcels. They have shown that their methodology can diminish the transmission capacity utilized by tests, improve the precision of estimation, and abatement the multifaceted nature of choosing ways or trees to send tests.
2. Martin Turon. **A Sensor Network Monitoring and Management Tool** - *Crossbow Technology, Inc*; This paper presents a adaptable programming system for overseeing, checking, and picturing sensor arrange organizations called MOTE-VIEW. In this paper, they directed a extensive audit of normal issues experienced while sending and regulating remote sensor systems. At that point address these issues by sketching out a observing instrument engineering utilizing an extensible arrangement of UI segments explicitly focused towards improving reasonability.
3. Baris Kurt, Engin Zeydan. **A Network Monitoring System for High Speed Network Traffic**. [Conference Paper · June 2016]: *Department of Computer Engineering, Bogazici University*; This paper comprehensively features observing system, measurements is significant for the support and framework making arrangements for the system specialist co-ops. In this exhibit, we will feature an underlying investigation of a universally useful system checking stage for rapid versatile systems. The created stage is the reason for performing complex continuous examination.
4. Travis Keshav. **A Survey of Network Performance Monitoring Tools** *Transport in Porous Media*, 90(2); In this day and age of systems, it isn't sufficient just to have a system; guaranteeing its ideal execution is critical. This paper examinations a few features of Network Performance Monitoring, assessing a few inspirations just as inspecting numerous business. In this paper, a few distinct parts of Network Performance Monitoring have been examined, with various proposed instruments for every theme; the arrangement no longer just must be to get more transfer speed or to overhaul the servers each time an issue is found.

## 3 Problem Statement

SNMP provides a utility to define and manage the network information and also for the information exchange. There is a requirement to build a management station which is going to monitor all the interactions in network with the help of SNMP agent and manager. The management system should be able to display the traffic at various devices. With increase in network devices and traffic, monitoring and management system is required to increase and assess the performance.



## 4 Objective

To design a system for network monitoring and perform analysis on the gathered data using Simple Network Management Protocol.

*“To visualize and perform analysis on network data.”*

## 5 Methodology

To achieve our objective, following steps need to be followed—

**Step 1:** Study of different networking features and identify dependent protocols.

**Step 2:** Study of SNMP trap and walk.

**Step 3:** Registering small network devices to gather information.

**Step 4:** Collection and Monitoring of network data.

**Step 5:** Perform and draw important information from data.

**Step 6:** Analysis of performance.

## 6 Design

The most important and critical component of any software product development is design. To design and develop this tool, object-oriented methodology is used. Object oriented models gives us a new insight on modelling real world entities into objects, it provides an immense way of thinking about problems using different object models. Object is the only way to keep data structure and behaviour together as a single entity. This model describes the structure of the objects their identity, their relationship with other objects, their attributes and their operations. Graphical notation is used for modelling objects and its classes, to provide the view of their relationships to one another. This notation is useful for abstract as well as actual problem designing.

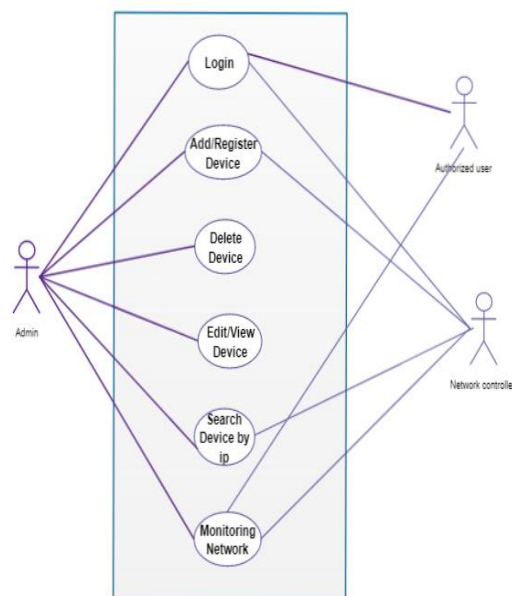


Figure 1: Use Case Diagram of network monitoring system

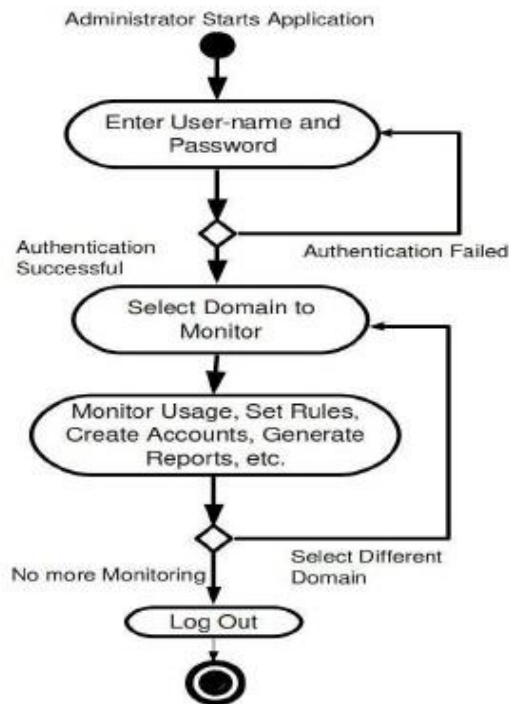


Figure 2: Activity Diagram of network monitoring system

## 7 Implementation

### 7.1 Pseudo Code

#### 7.1.1 Import necessary libraries

```

/**
 * Created by abhishek on 15/4/20.
 */
import java.awt.AWTException;
import java.awt.CheckboxMenuItem;
import java.awt.Image;
import java.awt.Menu;
import java.awt.MenuItem;
import java.awt.PopupMenu;
import java.awt.SystemTray;
import java.awt.TrayIcon;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.UnknownHostException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.text.SimpleDateFormat;
import java.util.Date;

```

Figure 3: Importing Necessary Libraries

### 7.1.2 Monitor Definition

```
public MonitorDefinition(){  
}  
  
public Monitor getMonitorInstance() throws Exception{  
    Monitor monitor = (Monitor)Class.forName(getClassName()).newInstance();  
    monitor.setMonitorDefinition(this);  
    return monitor;  
}
```

Figure 4: *Monitor Definition*

### 7.1.3 Monitor Thread

```
public MonitorThread(Monitor monitor){  
    this.monitor = monitor;  
}
```

Figure 5: *Monitor Thread*

### 7.1.4 Monitor Thread Run

```
public void run(){  
    monitor.setActivity(ActivityState.RUNNING);  
    LOOP: while(running){  
        monitor.setActivity(ActivityState.TESTING);  
        monitor.execute();  
        monitor.setActivity(ActivityState.RUNNING);  
  
        try{  
            Thread.sleep(monitor.getDelay());  
        } catch (InterruptedException e){  
            if(!running){  
                break LOOP;  
            }  
        }  
    }  
    monitor.setActivity(ActivityState.STOPPED);  
}
```

Figure 6: *Run Monitor*

### 7.1.5 Start Monitor

```
public synchronized void startMonitor(){  
    if(running){  
        return;  
    }  
  
    monitor.setActivity(ActivityState.STARTING);  
  
    running = true;  
    thread = new Thread(this);  
    thread.start();  
}
```

Figure 7: *Start Monitor*

### 7.1.6 Stop Monitor

```
public synchronized void stopMonitor(){
    if(!running){
        return;
    }

    monitor.setActivity(ActivityState.STOPPING);

    running = false;
    thread.interrupt();
}
```

Figure 8: *Stop Monitor*

### 7.1.7 Network Monitoring Request

```
public class JNMRequest{

    public JNMRequest(List data){
        this.data = new byte[data.size()];
        for(int i = 0; i < data.size(); i++){
            Byte b = (Byte)data.get(i);
            this.data[i] = b.byteValue();
        }
    }
}
```

Figure 9: *Network Monitoring Request*

### 7.1.8 Network Monitoring Response

```
public class JNMResponse{

    public JNMResponse(List data, long timeOfRequest){
        this.data = new byte[data.size()];
        for(int i = 0; i < data.size(); i++){
            Byte b = (Byte)data.get(i);
            this.data[i] = b.byteValue();
        }
        this.timeOfRequest = timeOfRequest;
    }
}
```

Figure 10: *Network Monitoring Response*

### 7.1.9 Test URL

```
while (processReturnValue != -1)
{
    try
    {
        URL siteURL = new URL(ip);
        connection = (URLConnection) siteURL.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();
        code = connection.getResponseCode();

        if (code >= 400)
            processReturnValue = 1;

        result = String.valueOf(code) + ": "
            + connection.getResponseMessage();
    }
}
```

Figure 11: *Test URL*

### 7.1.10 Test IP

```
private void testIP(String ipType, String ip, String timeout)
{
    Process p = null;
    int processReturnValue = 0;
    boolean wasLastStatusError = false;

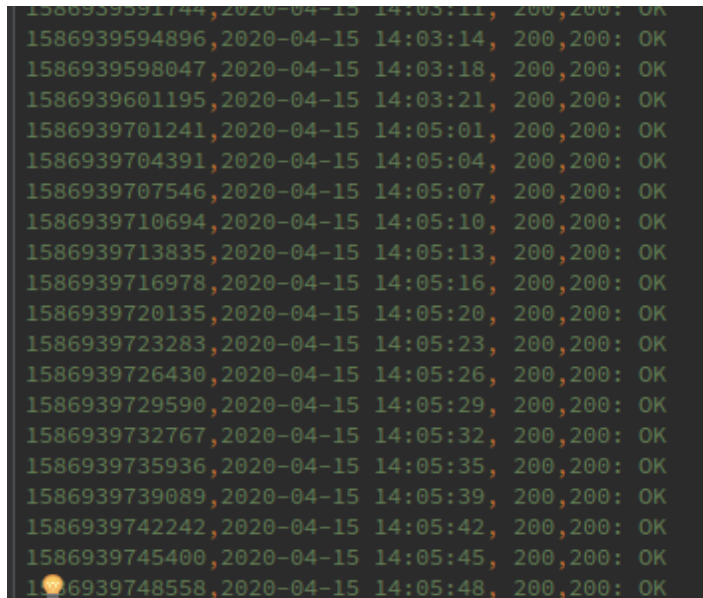
    while (processReturnValue != -1)
    {
        try
        {
            p = Runtime.getRuntime().exec(
                "ping -t -w " + timeout + " " + ip);

            BufferedReader inputStream = new BufferedReader(
                new InputStreamReader(p.getInputStream()));

            String pingMessage = "";
            // reading output stream of the command
            while ((pingMessage = inputStream.readLine()) != null)
```

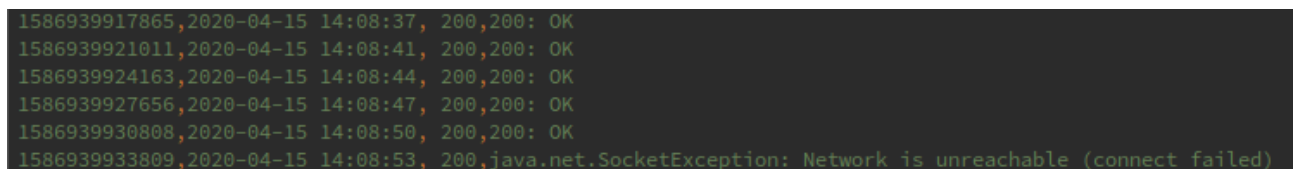
Figure 12: *Test IP*

## 7.2 Output Screen



```
158693991744,2020-04-15 14:03:11, 200,200: OK
1586939594896,2020-04-15 14:03:14, 200,200: OK
1586939598047,2020-04-15 14:03:18, 200,200: OK
1586939601195,2020-04-15 14:03:21, 200,200: OK
1586939701241,2020-04-15 14:05:01, 200,200: OK
1586939704391,2020-04-15 14:05:04, 200,200: OK
1586939707546,2020-04-15 14:05:07, 200,200: OK
1586939710694,2020-04-15 14:05:10, 200,200: OK
1586939713835,2020-04-15 14:05:13, 200,200: OK
1586939716978,2020-04-15 14:05:16, 200,200: OK
1586939720135,2020-04-15 14:05:20, 200,200: OK
1586939723283,2020-04-15 14:05:23, 200,200: OK
1586939726430,2020-04-15 14:05:26, 200,200: OK
1586939729590,2020-04-15 14:05:29, 200,200: OK
1586939732767,2020-04-15 14:05:32, 200,200: OK
1586939735936,2020-04-15 14:05:35, 200,200: OK
1586939739089,2020-04-15 14:05:39, 200,200: OK
1586939742242,2020-04-15 14:05:42, 200,200: OK
1586939745400,2020-04-15 14:05:45, 200,200: OK
1586939748558,2020-04-15 14:05:48, 200,200: OK
```

Figure 13: *Monitoring of URL when connected*



```
1586939917865,2020-04-15 14:08:37, 200,200: OK
1586939921011,2020-04-15 14:08:41, 200,200: OK
1586939924163,2020-04-15 14:08:44, 200,200: OK
1586939927656,2020-04-15 14:08:47, 200,200: OK
1586939930808,2020-04-15 14:08:50, 200,200: OK
1586939933809,2020-04-15 14:08:53, 200,java.net.SocketException: Network is unreachable (connect failed)
```

Figure 14: *Monitoring of URL when network is not connected*

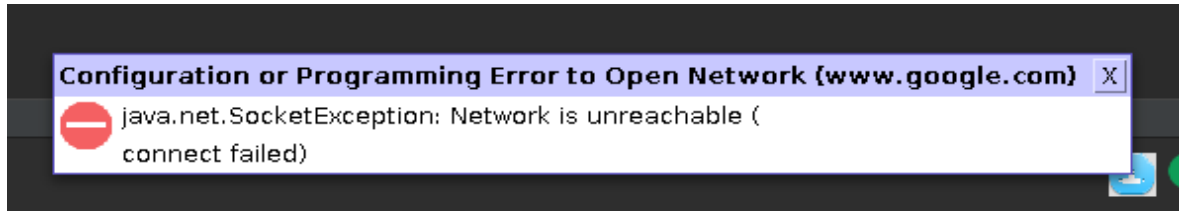


Figure 15: Alert when disconnected

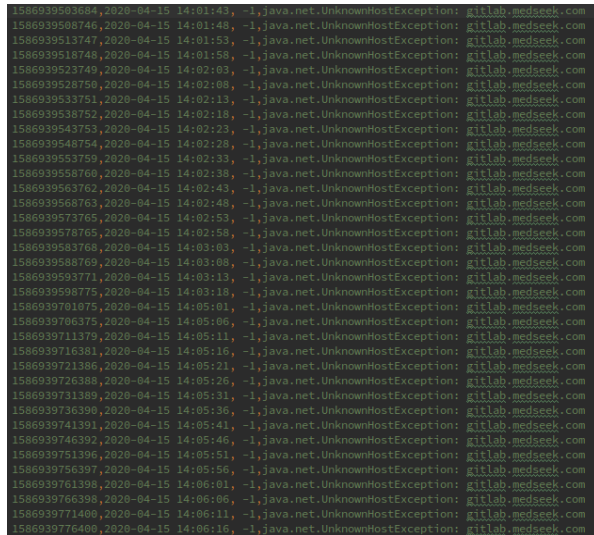


Figure 16: Monitoring of VPN



Figure 17: Alert when VPN is not present

### 7.3 Result Analysis

Let us consider a Local Area Network to understand the working of this project. In this network there is a switch, one router (Gateway) and some server attach to it to access the Internet. Now, step by step working of network monitoring system will be explained.

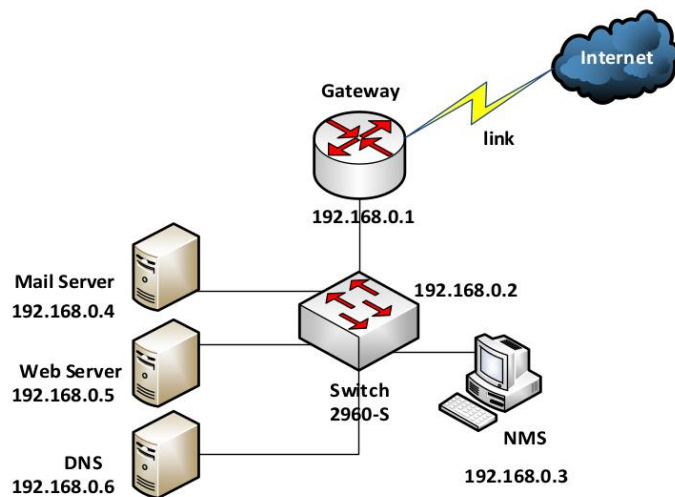


Figure 18: *Local Area Network*

### STEP 1: Checking the device availability

Using the ping command, NMS can detect the availability of network devices as well as hosts in the network.

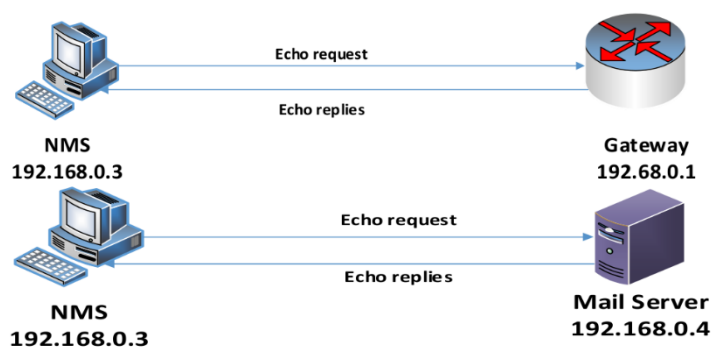


Figure 19: *Device Availability*

### STEP 2: Checking the availability of Services.

This project uses the port scanning method, to check the availability of services. It is been described in detailed in the below figure:

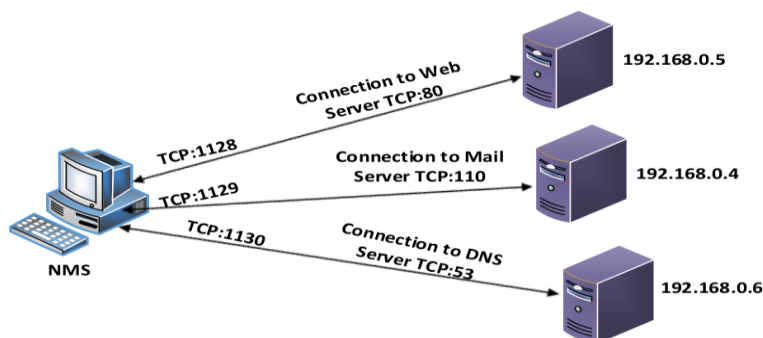


Figure 20: *Service Availability*

## **8 Conclusion and Future Scope**

This project was carried out to provide a management and maintenance system for the network administrator or organisation to look after the network traffic and bandwidth. This project provides a helping hand to identify what are the causes which are affecting the network. It can also be used to perform analysis on the network data so as to increase the performance of network. means by which the elderly and disabled in our society can use technology. This project was accomplished with the help of SNMP and ICMP protocols. In the coming future, demand and need of network monitoring system will increase as the network is increasing, so updates and alternative approach will enhance this project. User experience in this particular project shall be improved and roles may be defined for different users. In this project, a good AI based desktop assistant with less complex design and working can successfully be attached. Further this project can be used in various fields like in Telecoms industry to aid the development of better self-care services, in large network organisation to help and assist their network demands.



## References

- [1] Christina Fragouli, Athina Markopoulou (2005). **A Network Coding Approach to Overlay Network Monitoring**- *Stanford University*;
- [2] Martin Turon. **A Sensor Network Monitoring and Management Tool** - *Crossbow Technology, Inc.*;
- [3] Baris Kurt , Engin Zeydan. **A Network Monitoring System for High Speed Network Traffic**. [Conference Paper · June 2016]: *Department of Computer Engineering, Bogazici University*
- [4] Travis Keshav. **A Survey of Network Performance Monitoring Tools** *Transport in Porous Media*, 90(2);

## A APPENDIX I PROJECT CODE

### Module 1: Ping Task and Scheduler

```
public void connect(String ip) throws IOException {
try {
    String[] command = {"/bin/bash", "-c", "ping -i 2 -W 2 -t 2 -c 2 " + ip};
    ProcessBuilder build = new ProcessBuilder(command);
    Process process = build.start();
    input=newBufferedReader(newInputStreamReader(process.getInputStream()
    ));
    error=newBufferedReader(newInputStreamReader(process.getErrorStream()
    ));
    String output = null;
    System.out.println("Output : ");
    while ((output = input.readLine()) != null) {
        System.out.println(output);
    }
    if ((output = error.readLine()) != null) {
        System.out.println("Error, if any: ");
        while ((output = error.readLine()) != null) {
            System.out.println(output);
        }
    }
} } }
catch (IOException f)
{
    f.printStackTrace();
}
catch (Exception e)
{
    System.out.println(e);
}
finally {
    input.close();
    error.close();
} }

public class Scheduler extends Thread {
public void run() {
try {
    while (!(GlobalConfig.getState())) {
        HashMap<String, Integer> localconfiglog =
        GlobalConfig.getConfiguration();
        for (Map.Entry<String, Integer> entry : localconfiglog.entrySet()) {
            entry.setValue(entry.getValue() - 10);
            if (entry.getValue() == 0) {
                GlobalConfig.send(GlobalConfig.getkey(entry));
                entry.setValue(GlobalConfig.getvalu(GlobalConfig.getkey(entry)));
            }
        }
    }
}
}
```

```

    } } }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (!(GlobalConfig.getState())) {
        run();
    } } } }

```

## Module 2: Global Configuration where all data resides

```

public class GlobalConfig {
    public static HashMap<String, Integer> configuration = new HashMap<String,
    Integer>();
    public static HashMap<String, Integer> configlog = new HashMap<String,
    Integer>();
    public static BlockingQueue queue=new LinkedBlockingQueue();
    public static AtomicBoolean isShutdown=new AtomicBoolean(false);
    public static void store() throws FileNotFoundException {
    try {
        Yaml yml = new Yaml();
        String path = "/home/abhishek/IdeaProjects/MultiAssign2/samap.yml";
        configuration = yml.load(new FileInputStream(new File(path)));
        System.out.println("Records: ");
        System.out.println(configuration);
        configlog.putAll(configuration);
    }
    catch (FileNotFoundException f) {
        f.printStackTrace();
    }
    catch (Exception e)
    {
        System.out.println(e);
    }
    }
    public static HashMap<String,Integer> getConfiguration()
    {
        return configlog;
    }
    public static Integer getvalu(String key){

        return configuration.get(key);
    }
    public static boolean getState()
    {
        return isShutdown.get();
    }
    public static String getKey(Map.Entry<String,Integer> local)
    {

```

```

        return local.getKey();
    }
    public static Integer getvalue(Map.Entry<String,Integer> local)
    {
        return local.getValue();
    }
    public static void send(String key){
    try{
        queue.put(key);
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    }
    public static String receive() {
    String ip = null;
    try {
        ip = (String) queue.take();
    } catch (InterruptedException e) {
    e.printStackTrace();
    } catch (Exception e){
        System.out.println(e);
    }
    return ip;
    }
    }
}

```

### Module 3: Port Scanning Using Socket Programming

```

public class PortServer {
    public static void main(String[] args) throws IOException {
        try{
            System.out.println("Waiting for Clients: ... ");
            Integer totalNo=1;
            ServerSocket ss=new ServerSocket(9806);
            long time=0;
            while(time < 1000) {
                Socket soc = ss.accept();
                long t1=System.currentTimeMillis();
                BufferedReader in =new      BufferedReader(new
                InputStreamReader(soc.getInputStream()));
                FileWriter myWriter=new      new
                FileWriter("/home/abhishek/IdeaProjects/s3.txt",true);
                String s=in.readLine();
            }
        }
    }
}

```

```

        myWriter.write(s);
        myWriter.close();
        long t2=System.currentTimeMillis();
        time=time+t2-t1;
        System.out.println(time);
    }
    System.exit(0);
}
catch (Exception e)
{
    e.printStackTrace();
} } }

```

#### Module 4: Creating System Tray

private void createAndAddApplicationToSystemTray() throws IOException

```

{
    if (!SystemTray.isSupported())
    {
        System.out.println("SystemTray is not supported");
        return;
    }

    final PopupMenu popup = new PopupMenu();
    ClassLoader classLoader = Thread.currentThread()
        .getContextClassLoader();
    InputStream inputStream = classLoader
        .getResourceAsStream("netmon.png");
    Image img = ImageIO.read(inputStream);

    final TrayIcon trayIcon = new TrayIcon(img, TOOL_TIP);
    this.processTrayIcon = trayIcon;
    final SystemTray tray = SystemTray.getSystemTray();
    MenuItem aboutItem = new MenuItem("About");

    CheckboxMenuItem autoSizeCheckBox = new CheckboxMenuItem(
        "Set auto size");
    CheckboxMenuItem toolTipCheckBox = new CheckboxMenuItem("Set
tooltip");

    Menu displayMenu = new Menu("Display");

    MenuItem errorItem = new MenuItem("Error");
    MenuItem warningItem = new MenuItem("Warning");
    MenuItem infoItem = new MenuItem("Info");
    MenuItem noneItem = new MenuItem("None");

    MenuItem exitItem = new MenuItem("Exit");

```

```
popup.add(aboutItem);
popup.addSeparator();
popup.add(autoSizeCheckBox);
popup.add(toolTipCheckBox);
popup.addSeparator();
popup.add(displayMenu);
displayMenu.add(errorItem);
displayMenu.add(warningItem);
displayMenu.add(infoItem);
displayMenu.add(noneItem);
popup.add(exitItem);
```

```
trayIcon.setPopupMenu(popup);
toolTipCheckBox.setState(true);
autoSizeCheckBox.setState(true);
trayIcon.setImageAutoSize(true);
```

```
try
{
    tray.add(trayIcon);
}
catch (AWTException e)
{
    System.out.println("TrayIcon could not be added.");
    return;
}
trayIcon.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(
            null,
            "This dialog box is run from System Tray");
    }
});
aboutItem.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(
            null,
            "This dialog box is run from the About menu item");
    }
});
autoSizeCheckBox.addItemListener(new ItemListener()
```

```

{
    public void itemStateChanged(ItemEvent e)
    {
        int autoSizeCheckBoxId = e.getStateChange();
        if (autoSizeCheckBoxId == ItemEvent.SELECTED)
        {
            trayIcon.setImageAutoSize(true);
        }
        else
        {
            trayIcon.setImageAutoSize(false);
        }
    }
});
toolTipCheckBox.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        int toolTipCheckBoxId = e.getStateChange();
        if (toolTipCheckBoxId == ItemEvent.SELECTED)
        {
            trayIcon.setToolTip(TOOL_TIP);
        }
        else
        {
            trayIcon.setToolTip(null);
        }
    }
});
ActionListener listener = new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        MenuItem item = (MenuItem) e.getSource();
        System.out.println(item.getLabel());
        if ("Error".equals(item.getLabel()))
        {
            trayIcon.displayMessage(
                MESSAGE_HEADER,
                "This is an error message",
                TrayIcon.MessageType.ERROR);
        }
        else if ("Warning".equals(item.getLabel()))
        {
            trayIcon.displayMessage(
                MESSAGE_HEADER,
                "This is a warning message",

```

```

        TrayIcon.MessageType.WARNING);
    }
    else if ("Info".equals(item.getLabel()))
    {
        trayIcon.displayMessage(
            MESSAGE_HEADER,
            "This is an info message",
            TrayIcon.MessageType.INFO);
    }
    else if ("None".equals(item.getLabel()))
    {
        trayIcon.displayMessage(
            MESSAGE_HEADER,
            "This is an ordinary message",
            TrayIcon.MessageType.NONE);
    }
}
};
errorItem.addActionListener(listener);
warningItem.addActionListener(listener);
infoItem.addActionListener(listener);
noneItem.addActionListener(listener);
exitItem.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        tray.remove(trayIcon);
        System.exit(0);
    }
});
}

```

#### Module 5: Starting Process, Monitoring and Testing URL

```

private void startProcess()
{
    for (int ipCount = 0; ipCount < ipAddresses.length; ipCount++)
    {
        final int index = ipCount;
        Thread thread = new Thread(new Runnable()
        {
            @Override
            public void run()
            {
                monitorNetwork(
                    ipAddresses[index][0],
                    ipAddresses[index][1],
                    ipAddresses[index][2]);
            }
        });
    }
}

```



```

    });

    thread.start();
}
}

private void testingURL(String ipType, String url, String timeout)
{
    String result = "";
    int processReturnValue = 0;
    boolean wasLastStatusError = false;
    HttpURLConnection connection = null;
    int code = -1;

    String ip = url.startsWith("http://") ? url : ("http://" + url);

    while (processReturnValue != -1)
    {
        try
        {
            URL siteURL = new URL(ip);
            connection = (HttpURLConnection) siteURL.openConnection();
            connection.setRequestMethod("GET");
            connection.connect();
            code = connection.getResponseCode();

            if (code >= 400)
                processReturnValue = 1;

            result = String.valueOf(code) + ": "
                + connection.getResponseMessage();
        }
        catch (UnknownHostException | SocketTimeoutException e)
        {
            result = e.toString();
            processReturnValue = 1;
        }
        catch (Exception e)
        {
            result = e.toString();
            processReturnValue = -1;
        }
        finally
        {
            if (connection != null)
            {
                connection.disconnect();
            }
        }
    }
}

```

```

        connection = null;
    }
    try
    {
        logStatus(
            url,
            ipType,
            String.valueOf(System.currentTimeMillis()),
            String.valueOf(code),
            result);

        Thread.sleep(Long.parseLong(timeout));
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

switch (processReturnValue)
{
    case 0: // No error
        if (wasLastStatusError)
        {
            processTrayIcon.displayMessage(
                "Network Connection Established to " + ipType
                + " Network (" + url + ")",
                result,
                TrayIcon.MessageType.INFO);

            wasLastStatusError = false;
        }
        break;
    case 1: // Connection error
        if (!wasLastStatusError)
        {
            processTrayIcon.displayMessage(
                "Network Connection Error to " + ipType
                + " Network (" + url + ")",
                result,
                TrayIcon.MessageType.ERROR);

            wasLastStatusError = true;
        }
        break;
    case -1: // Configuration or programming error
        if (!wasLastStatusError)

```

```

        {
            processTrayIcon.displayMessage(
                "Configuration or Programming Error to "
                + ipType + " Network (" + url + ")",
                result,
                TrayIcon.MessageType.ERROR);

            wasLastStatusError = true;
        }
        break;
    }
}
}

```

### Module 6: Storing Log Details

```

private void logStatus(
    String ip,
    String ipType,
    String timeStamp,
    String messageType,
    String message) throws IOException
{
    Files.write(
        Paths.get(ipType + "." + ip + ".csv"),
        (timeStamp
            + ","
            + new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
                .format(new Date(Long.parseLong(timeStamp)))
            + ", " + messageType + ", " + message + "\n").getBytes(),
        StandardOpenOption.CREATE,
        StandardOpenOption.APPEND);
}

private static String[][] ipAddresses =
{
    {
        "Open", "www.google.com", "3000"
    },
    {
        "VPN", "gitlab.medseek.com", "5000"
    }
};

private static final String[][] pingResult =
{
    {
        "Error", "Request timed out",
        "The network connection may be unavailable"
    }
}

```

```

    },
    {
        "Error", "Ping request could not find",
        "A VPN may not be connected or established"
    },
    {
        "Wait", "Pinging",
        "Detecting network connection availability"
    },
    {
        "Success", "Reply from", "Network connection available"
    }
};

```

```

private boolean usePing = false;
private String TOOL_TIP = "Blue Atoll Network Monitor";
private String MESSAGE_HEADER = "Blue Atoll Network Monitor";
private TrayIcon processTrayIcon = null;
}

```

#### Module 7: Main Application or Boot Manager

```

public class NetworkMonitor
{
    public static void main(String[] args)
    {
        try
        {
            NetworkMonitor networkMon = new NetworkMonitor();
            networkMon.createAddSystemTray();
            networkMon.startProcess();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}

```