DESIGNING A SNMP MANAGER GUI BASED ON MONO FOR LINUX AND WINDOWS

Daniela – Roxana Soare, Valeriu Manuel Ionescu
University of Pitești, Romania
Faculty of Electronics, Communications and Computers
valeriu.ionescu@upit.ro

Keywords: Simple Network Management Protocol (SNMP), Mono, GUI, cross-platform

Abstract: *Creating charts and graphics plays a very important role in every graphic application as it makes data easier to read and understand. With the exception of web applications, creating cross platform Graphical User Interface (GUI) applications is more difficult as few libraries are supported across operating systems. The GUI application presented in this paper was designed using the Mono libraries to work on both Linux and Windows operating systems for asking and handling the information received from a Simple Network Management Protocol (SNMP) agent*.

1. INTRODUCTION

Mono is a development platform available on Windows and Linux that uses C# programming language and provides both the computational capabilities of generating data as a simulation engine and displaying it in a variety of graphical representations based on its Graphical Device Interface (GDI+).

The power of the C# programming language, combined with simplicity of implementing a Form application in Mono, makes real-world program development faster and easier than ever before.

Using the development environment for Mono, MonoDevelop [1,2], and the C# programming language, in this application were made a variety of XY two-dimensional graphics realised using basic fundamental graphic objects such as lines, rectangles, circles, ellipses or polygons organized in a easy to read and interpret manner.

The data that will be charted by this application is offered by network SNMP agents. SNMP [3] is an application layer protocol that is used for collecting data about a device's operation ranging from CPU usage, encountered errors to network operation.

A SNMP system consists of three elements:

- An agent that collects data about a managed device where it is installed. Usually each characteristic of the device is refreshed at a specific rate. This means that for the different aspects that are interrogated, the updated values will appear at different time intervals (ranging from seconds to minutes). Usually agents respond to requests but can also send information without a request if a specific condition was met.
- A network manager running on a network management station that queries the agents for information.
- A Mangement Information Database is a hierarchy that describes the variables accessible via SNMP. MIBs contain the object identifiers (OID) that identify a variable that can be read or set via SNMP.

The paper is organized as follows:

- In chapter 2 the application architecture is presented;
- In chapter 3 the design of the different charts types implemented is explained and their implementation is discussed;
- In chapter 4 the application is tested and various implemented graphics are presented, in both Linux and Windows operating systems, making a comparison of the resulting differences;
- Finally, in the last chapter conclusions are presented and considerations for future implementations.

2. APPLICATION ARCHITECTURE

The application presented is the network manager that will send requests to the agents in the network. The agents can be network hardware with SNMP support or network computers.

The system that will be interrogated receives the request chosen in the GUI interface and returns information about its performances that will be later taken and transposed into charts.

The traffic between the monitor system and the agent is presented in Fig. 1.

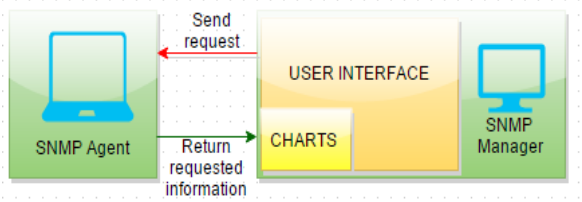


Fig. 1. Functional blocks that shows application flow

The user (called the client) can perform the actions presented in Fig. 2.

The client/user must know only the IP of the managed network device. For a better visualisation of the computer's performances, the application provides the option of choosing a type of performance which will be displayed on a different type of chart.

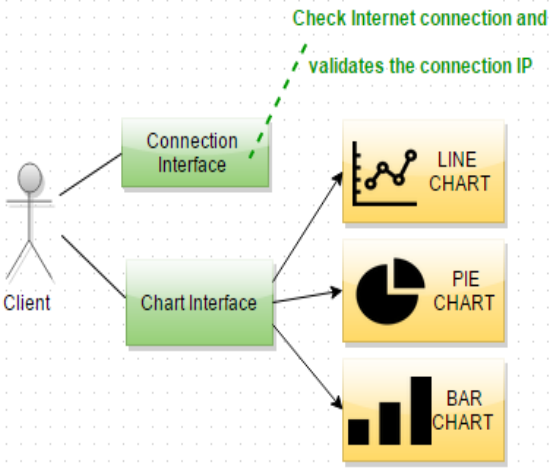


Fig. 2. Utilization case diagram for Client

As a computer can have multiple interfaces, an interface selection menu will be presented once, when the application starts. It checks for the Internet connection to be available and when it's established, the user can type the IP of the device he wants to connect to, following the steps from the diagram in Fig. 3.

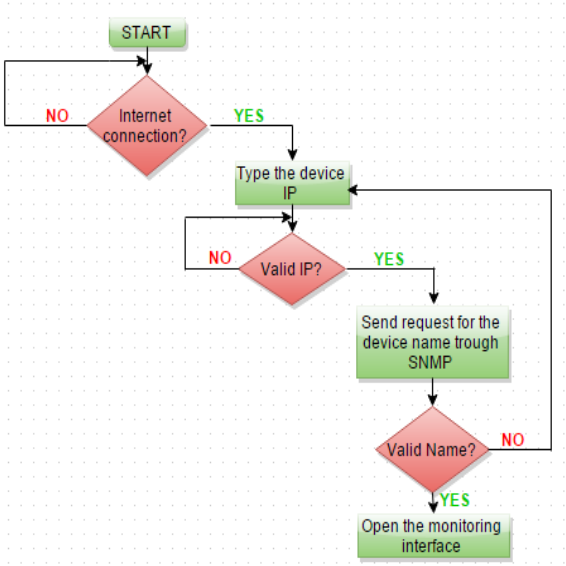


Fig. 3. Activities diagram of the Connection Interface

While the Chart Interface is visible, a process is run continuously in the background in order to get and refresh the display new data from the agents, with a speed depending on user selection.

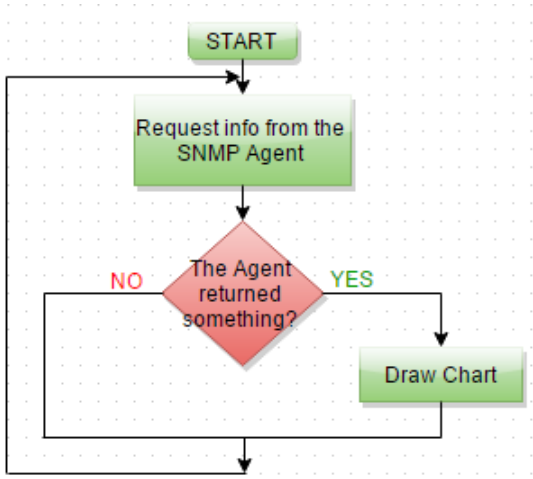


Fig. 4. Activities diagram of the Chart Interface

a. line chart design

In order to create the line chart, the code was divided into four basic structures to be easily extended and to be able to add new features: *DataSeries*, *DataCollection*, *ChartStyle* and *LineStyle*.

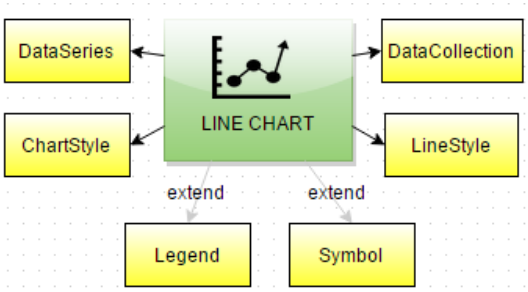


Fig. 5. Line chart organization

The chart style structure includes all chart layout related information. The data series holds the chart data and the line styles.

Data collection holds the data series, with each data series representing one curve on the chart. The line style structure is used to specify the line colour, thickness, dash style etc.

For a 2D X-Y chart with multiple lines, the user may want to use a legend to identify each line plotted on the chart. This legend shows a sample of the line type, colour and text label the user specifies.

Also, the user might want the chart to display not only lines but the symbols as well at the data points, or might want a chart with only symbols and no lines. That's why the *Symbol* and the *Legend* structure were added as an extension of primary line chart.

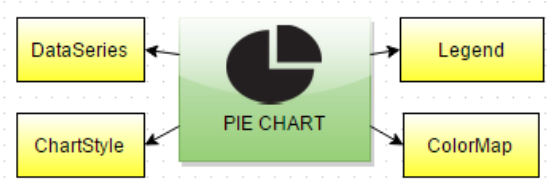


Fig. 6. Pie chart organization

b. pie chart design

The pie chart was divided into four main structures, like in the previous chart design, except that instead of *DataCollection* structure we have one for the *ColorMap*.

The *ColorMap* structure defines a mapping between existing colours and the new colour to which they are to be converted. When the map is applied, any pixel of the old colour is converted to the new colour. These colour maps are simply tables or lists of colours that are organized in some desired fashion. The user can easily create a color map with an *m x 4* color map matrix. Each row of the matrix represents ARGB values. The row index represents the y data of a 2D chart. For a given colour map matrix with *m* rows, the colour data values can be linearly scaled to the colour map.

For example, if the user wants to use the colour map to represent the y coordinates of a 2D graphics object, it can use the *YMin* and *YMax* to linearly transform the y data values to indices where each index identifies an ARGB row (i.e., a colour) in the colour map matrix. The mathematical transformation of the colour index values is described by the formula:

$$Color\ Index = \begin{cases} 1 & y < Y\min \\ (\text{int})\left(\dfrac{(y - Y\min)m}{Y\max - Y\min}\right) & Y\min \le y < Y\max \\ m & y \ge Y\max \end{cases}$$

Here y is the individual value of the *Y* data and *m* is the length of the color map matrix. This allows the user to use the entire range of colors in the color map over the plotted data.

c. bar chart design

The bar chart is useful for comparing groups of data. In bar chart, a group can have a single category of data, or can be broken down further into multiple categories for a greater depth of analysis.

A bar chart is often used in exploratory data analysis to illustrate the major features of the distribution of the data in a convenient form. It displays the data using a number of rectangles of the same width, each of which represents a particular category.

The length (and the hence area) of each rectangle is proportional to the number of cases in the category it represents.

As organization, this kind of chart has the following structure:

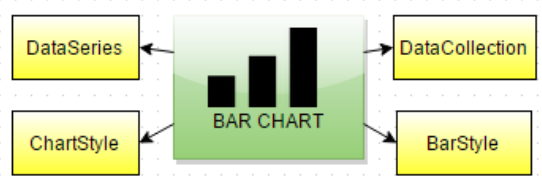


Fig. 7. Bar chart organization

DataSeries, *DataCollection* and *ChartStyle* structures are the same as those used for the line chart and the forth one (*BarStyle*) is used to specify the bars corresponding properties like bar width, fill color and border style.

If the line chart drawing is based on the plot rectangle viewport, the pie and bar charts are based on a drawing panel. The advantage of the first viewport is that it is simple, allowing you to draw both the chart style (tick labels, axis labels and title) and the data curves on the same form. Its drawback is that the user have to deal with the possibility of drawing outside of the viewport when the limits of the axes changes.

For the second kind of viewport, the user can draw anything inside it. It doesn't need to worry about drawing outside the viewport. Since the bar chart and the pie chart involve drawing rectangles or polygons, this viewport will be a natural choice.

The implementation of the Barchart class is as follows:

```
public class BarStyle
    {
        private Color fillColor = Color.Black;
        private Color borderColor = Color.Black;
        private float borderThickness = 1.0f;
        private float barWidth = 0.8f;
        private DashStyle borderPattern =
DashStyle.Solid;
        public float BarWidth
        {
            get { return barWidth; }
            set { barWidth = value; }
        }
        virtual public DashStyle BorderPattern
        {
            get { return borderPattern; }
            set { borderPattern = value; }
        }
        public float BorderThickness
        {
            get { return borderThickness; }
            set { borderThickness = value; }
        }
        virtual public Color FillColor
        {
            get { return fillColor; }
            set { fillColor = value; }
        }
        virtual public Color BorderColor
        {
            get { return borderColor; }
            set { borderColor = value; }
        }
    }
```

The libraries used are:

```
using System.Drawing;
using System.Drawing.Drawing2D;
```

d. Other application elements

The **connection interface** consists of four main functions and one timer type that will check the internet connection at an interval of 5 seconds for any unexpected shutdowns.

The bool function named *CheckForInternetConnection* has the duty of ensuring the existence of internet connection by checking data received from an identified resource, in our case the Google website.

Inside *CheckIpAddress* function it's validated the correctness of the introduced IP connection. It splits the IP string after point and checks if each group of figures is in the range 0-255.

Another function is *CheckIfCorrect*, sends a request to the SNMP Agent for the device name and when the response comes back it displays it in a MessageBox on the screen so the user will confirm whether is the right device name or not.

When the client validates the device name it will open another form (**Chart Interface**) witch will display some of the performances of the interrogated device.

One method *CPUUsage* will run cyclically at a fixed interval of one second and it sends on the console a request to get the CPU usage of the device. For memory usage it calls the *MemUsage* method that reruns at the same period of time.

The CPU usage request can be made for 1, 5 or 10 minutes. For in application it was chosen the 1 minute refresh interval.

The memory usage was calculated sending three SNMP requests: *StorageUsed*, *AllocUnits* and *StorageSize*.

The formula used is:

*Total memory used = StorageUsed * AllocUnits;*
*Total memory = StorageSize * AllocUnits;*

To get the Internet traffic the request needed to be made twice, at an interval of one second. The result was obtained by subtracting the first value obtained from the second one.

3. APPLICATION TESTING

In order to have an easy to use and repeatable testing environment giving conclusive results, virtual machines were used as agents, and the manager was run on real Windows and Linux computers.

The application that will be used as a support for SNMP protocol communication is snmpwalk that has both Windows and Linux binaries included in Net-SNMP [4].

In order to get the requested information from the device the OID list presented in Table 1 was used.

Table 1. OID list used in application development

OID	Meaning
1.3.6.1.2.1.25.1.1.0	System Uptime
1.3.6.1.2.1.25.2.3.1.6	Storage Used
1.3.6.1.2.1.25.2.3.1.4	Allocation Units
1.3.6.1.2.1.25.2.3.1.5	Storage Size
1.3.6.1.2.1.2.2.1.10	Outgoing Bytes
1.3.6.1.2.1.2.2.1.16	Incoming Bytes
1.3.6.1.2.1.25.3.3.1.2	CPU Usage

After the first implementing of the line chart, the result is visible in Fig. 8.

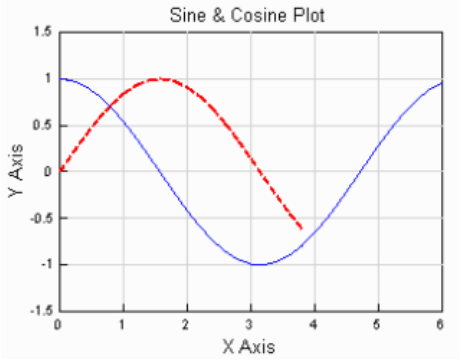


Fig. 8. Sine and Cosine line chart

For this kind of chart, if the user enables the legend, the result is visible in Fig. 9.

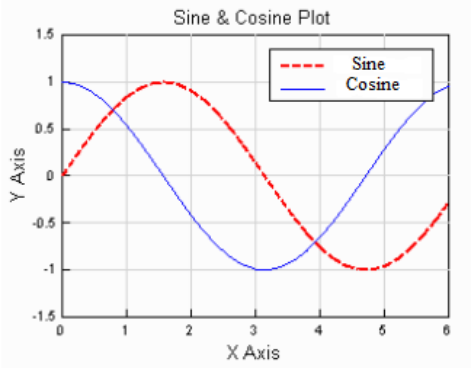


Fig. 9. Sine and Cosine line chart with legend

Also, if the user adds some symbols to the chart, the result is visible in Fig. 10.

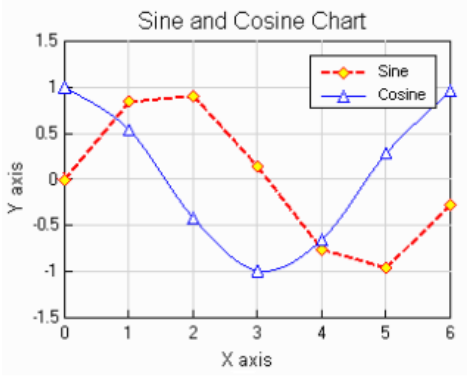


Fig. 10. Sine and Cosine line chart with legend and symbols

In this application I have used the line chart with one data series (CPU usage) and with legend. The user can change the line style and also add symbols to it.

After implementing the pie chart is visible in Fig. 11.

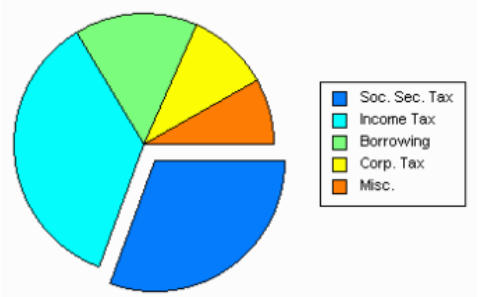


Fig. 11. Pie chart with an exploded piece

Using this kind of chart I've chose to display percentage of the used memory from the total of the memory.

The result of implementing the last type of chart is visible in Fig. 12. This type of chart was used to display the internet received and send traffic.

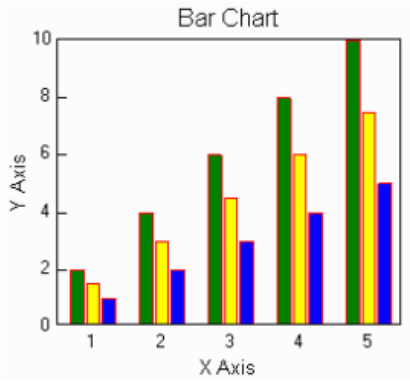
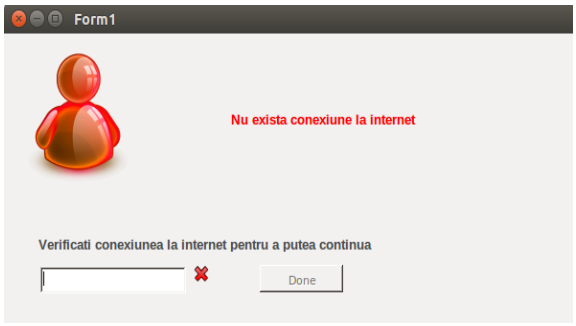
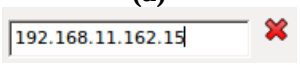


Fig. 12. Grouped vertical bar chart

If user wants to see the performance of a SNMP capable network device, the first step is to check the presence of the internet connectivity nor the network manager. If there is no internet connectivity, the application will display the message "No internet connection" and the user image will be red as seen in Fig. 13.a. In this case the user will not be able to connect to the device via SNMP.



(a)



(b)

Fig. 13. Application checks for (a) internet connection situation and (b) correct IP format

Once the connection will be restored and is available, the user can type the device IP which will be interrogated.

If the IP address was not entered correctly, it will appear next to the field a small "X" that warns the user about this (Fig. 13.b).

When the IP is valid, the image next to it will be no longer red and the button will be enabled so that it can be send the command to the SNMP Agent.

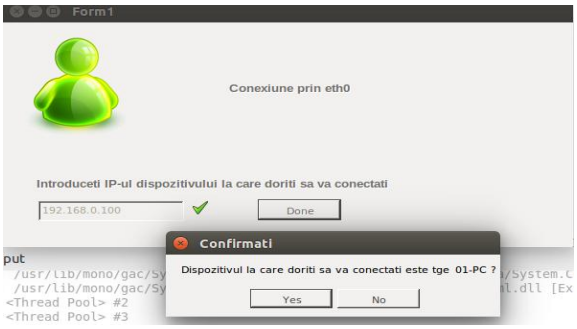


Fig. 14. Internet connection established and the requested device name

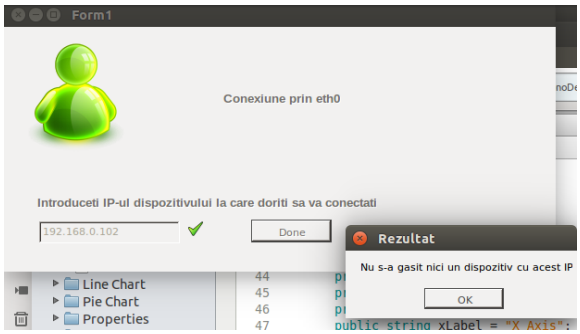


Fig. 16. Internet connection present but no agent found

If SNMP does not find any agent with the typed IP address then it will send back the following message: "Couldn't find any device with this IP".

After the response validation it will open a new form that will display the CPU performance information of the interrogated device.

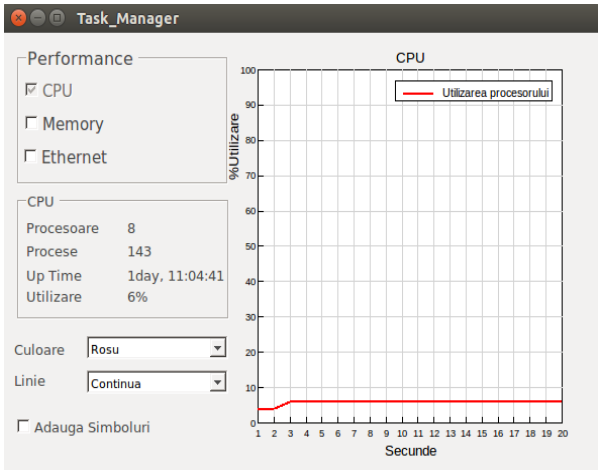


Fig. 17. CPU performance form

In this form the CPU activity is evidenced by a line chart that redraws the line at every second. As a display method, it is made up of a continuous red line.

On the left were highlighted the number of computer processors and the number of processes that are running on it. In order to highlight the effect of refreshing at every second I've added the Up Time field, which is constantly changing.

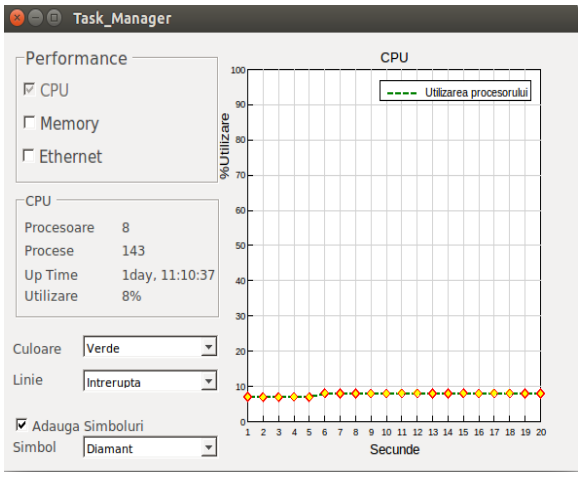


Fig. 18. CPU performance dashed green line, with diamond symbol

At this moment the chart appearance is changed according to the user preferences, as can be seen in the Fig. 18. The line of the chat was changed to one consisting of green dotted line (this change can be also seen in the legend), and for every point were added a diamond symbol.

When the user clicks on the Memory check box, the current form will close and will open a new one that will show memory performance in a pie chart.

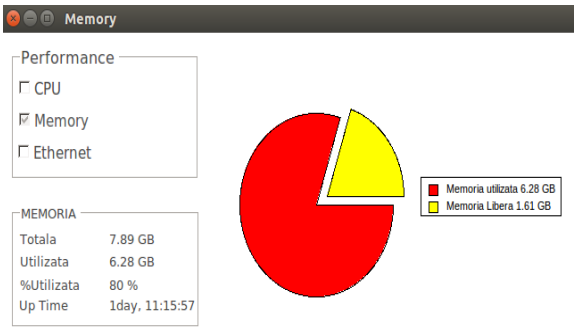


Fig. 19. Memory performance

In this form were highlighted the used memory percent in total memory. Also, at the level of this performance was also recorded the Up Time so that the user can see that the chart redraws at every second.

To view information about internet traffic the user will only need to click on the Internet check box, which will close the current form making room for the one with bar chart.

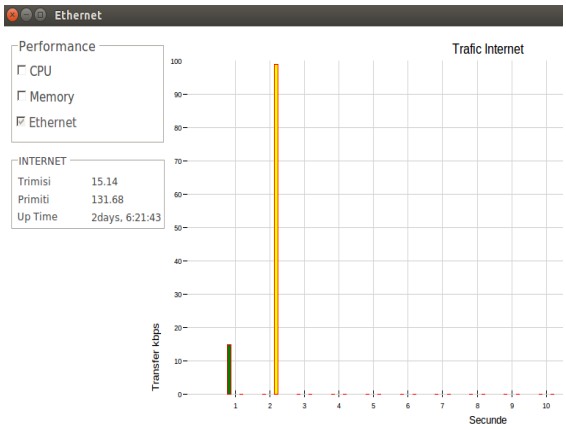
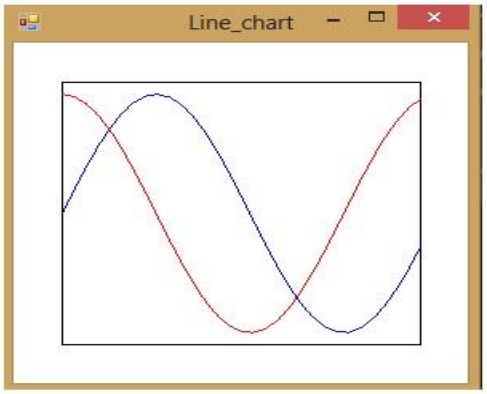
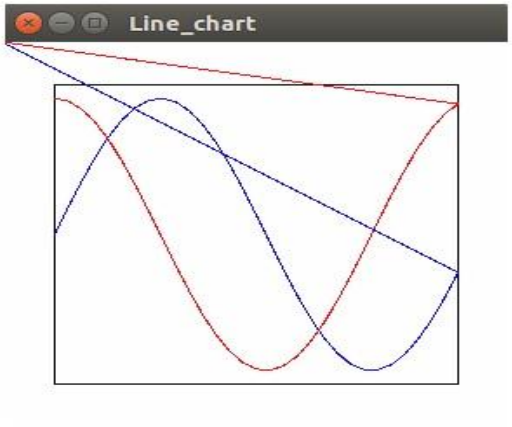


Fig. 20. Internet traffic chart

There were multiple problems encountered when developing the cross platform GUI. For example when Visual Studio was used to test the Windows implementation, for the same code of the line chart in Windows and in Linux, there were errors in the Linux display.



(a)



(b)

Fig. 21. The same line graphic in (a) Windows (Visual Studio 2013) and (b) Linux (MonoDevelop 5.5)

The solution was to omit the final drawing point of the graphic as not to coincide with the drawing area boundaries.

4. CONCLUSIONS

Monitoring the performance of a computer is a useful step for the future administration of server architectures. Because SNMP should be used from any device that the administrator can access, most applications are web applications, where the user interface is in HTML format.

There are many GUI SNMP manager applications that target either Windows or Linux. Those that target both usually offer a basic interface for free and a paid version for the full GUI.

This paper shows that it is possible to implement a GUI that can run on both Linux and Windows in order to display the graphics required by a SNMP manager.

The difficulties in developing this application were related to the file access (that is OS dependent) and minimal behaviour differences between Mono under Windows and Linux.

This SNMP manager GUI can be improved in the future with sending multiple interrogations in the background and only displaying the ones that the user chooses to be visible, and by creating a GUI for a statistics report based on the data stored locally from the interrogations.

5. BIBLIOGRAPHY

[1] MonoDevelop Project, Creating a simple user interface with MonoDevelop, accessed 06.06.2015, available at: http://www.monodevelop.com/documentation/creating-a-simple-user-interface-with-monodevelop/

[2] Niel M. Bornstein, Edd Dumbill, Mono: A Developer's Notebook, Ed. O'Reilly 2004

[3] RFC 3413 Internet Standard, Simple Network Management Protocol (SNMP) Applications, available at: https://tools.ietf.org/html/rfc3413

[4] Carnegie Mellon University, Net-SNMP - SNMP application configuration manual pages, available at http://net-snmp.sourceforge.net/