A .Net Framework Approach for a Network Monitoring Tool

Islam A.T.F. Taj-Eddin
The British University in Egypt (BUE)
Faculty of Informatics and Computer Science
El Sherouk City, Misr-Ismalia Road, Cairo - Egypt

ABSTRACT

Tools that monitor the performance of a PC are an essential part of today's quality control of the computer systems. Such resources needed to be monitored are the Memory, the CPU, the Power Consumption and the Network. Network Monitoring tools aim to audit and keep track of certain parameters to ensure the network's efficiency and to provide information that will assist in the network analysis. Network Monitoring tools are usually costly and not flexible enough to be adapted to the needs of the user. The problem of having a flexible, free and modern, in respect to the common industry knowledge, Network Monitoring tool is the goal of this paper. The solution aims to be a .Net framework open source product that is flexible to be adapted to the user's needs. It can be used by a single PC to monitor that PC's network or can be installed on a gateway PC to monitor the entire network. The proposed .Net solution is comparable with commercial and open source solutions. The comparison will show that the proposed .Net framework open source solution contains extra features that do not exist in neither commercial nor open source solutions.

Keywords

Computer networks, Monitoring tools, Network management, network monitoring system (NMS), Net Porter, NetLimiter, Wireshark, .Net Framework, C#, MS Visual Studio, regression test, Promiscuous mode.

1. INTRODUCTION

A network monitoring system (NMS) is the gatekeeper for the network. The main responsibilities of such system can be summarized as checking incoming and outgoing connections; that is it keeps an eye out for what might be going on within the network. There are different variables that can be monitored within a network, each capable of providing different performance/security metrics of its own. The latency for example, is a good way to discover failed and congested links, while the bandwidth would be a good parameter if we were to monitor suspicious networking activities, like denial of service. While the latency and bandwidth would be the first two parameters that come to mind when seeing the phrase *network monitoring*, there are more parameters such as packets dropped and retransmission related parameters. In addition, network monitoring can also include other non-network parameters, such as processor utilization, memory consumption and other performance metrics that might not be incorporated under the network monitoring category, but can be implemented as additional monitoring metrics to keep an eye out for other types of node/link failure whether the operating environment is a large network or a single PC [13].

A network monitoring application is not simply limited to watching the network in operation and checking for failed links/nodes; it can also help resolve some problems such as slow/lagged connections, lost data, overload and other similar network performance scalars that can provide information regarding any degraded performance for any connection (whether the solution is internal or external). Although there is some difference between an intrusion detection system (IDS), Intrusion prevention system (IPS) and a network monitor, we can tell that any IDS/IPS need to incorporate some sort of network monitoring device and/or software that would help in detecting or predicting any suspicious behavior [13].

Another critical outcome from monitoring the network is value-cost analysis for the network equipment; by identifying areas of heavy traffic and areas that need lower latency and increased bandwidth, user can easily identify what areas need upgrading; or in case overpriced network devices were used and the end node was not fully utilizing the equipment, the equipment would be downgraded or relocated to cut cost or provide the extra resources were it is actually needed, for a personal user it would provide the dimensions of a network interface card if the user plans to buy, upgrade and/or replace hardware and/or software resources.

The paper contribution is:

1- To propose a modern, flexible, open-source .Net network monitoring tool.
2- The References that will be used to develop the proposed open-source .Net network monitoring tool will be coming mainly from the common knowledge of the IT industry.

The proposed tool shall be named *Net Porter*. Providing such service for free in the form of open-source software will be more cost-effective. It will also provide more flexibility as the application can be customized in a more specific and less generic manner for the user of a .Net framework. The Net Porter can be used by a single PC to monitor that PC's network or can be installed on a gateway PC to monitor an entire network. In addition to that, remote reporting could be added since the software can be edited by the user.

A porter is a person who has charge of a door or gate; doorkeeper; a porter keeps track of who's coming in or out, who belongs to the building and who doesn't and of course may allow or disallow certain people from coming in. Security and the wellbeing of the residents is the main concern for the porter; a porter or a doorman's activities include-and are not limited to- knowing people in the building, confirming a guest's arrival with one of the tenants, and signing for packages [5].

The Net Porter is the PC's doorman. It displays all processes that are communicating on a network card and displays the

IPs associated with those connections along with numerical figures such as the data sent/received, upload/download speed…etc. and log any traffic associated with those IP in a log file along with a statistical log that provides a brief summary of the session. Just like a real porter, the Net Porter can prevent certain connections from communicating with the machine by killing the whole connection or certain parts of that connection.

The Proposed tool relies on three main tools, among many tools and techniques, to create a simple yet extensive network monitoring tool:

1- The .Net framework, using MS visual studio's C# , look to Figure 1 [11]:

The C# is a language that was developed by Microsoft that has many advantages over other programming languages; the first reason is the simplicity of the language, the C# is a relatively simple programming language that can easily be learned, and is similar to C, C++ and java in syntax. In addition to that, C# is an object oriented programming language. Encapsulation was a very useful feature of the language in this project. The main object *connection*, look to Table 4, has saved a lot of time in terms of development in debugging that would've been needed in developing multidimensional arrays and computationally linked arrays of different object types. Furthermore, C# provides interaction with native win32 dlls which basically means that any native win32 function that is not natively available for the C# could be called, which expands the functionality of the language to a near equivalent of the more versatile C++/C.

The language also offers a type safe development environment which might prove useful for future development. A type safe language would generally not allow the improper use of pointers that would complicate and halt execution, which leads to another feature, pointers. Pointers have been an important feature that has been used in the Net Porter, pointers are basically memory address holders which are very useful in case of unmanaged data types [30] (enums in case of the Net Porter), these memory address holders also evade garbage collection. Garbage collection is a memory management technique which frees unnecessarily consumed memory space; evading garbage collection is good when it is constantly needed to refer to an object of the same type for the same process/function. Type safety can be evaded by using the unsafe pointer feature of the .Net.

Finally, the c# has an amazing amount of libraries and wrappers that are responsible for performing almost any function that might be needed in a matter of a few lines of code. In addition to that, there are libraries that are provided with the .Net framework that make a programmer's life easier and manage system I/O, networking and security with minimal efforts.

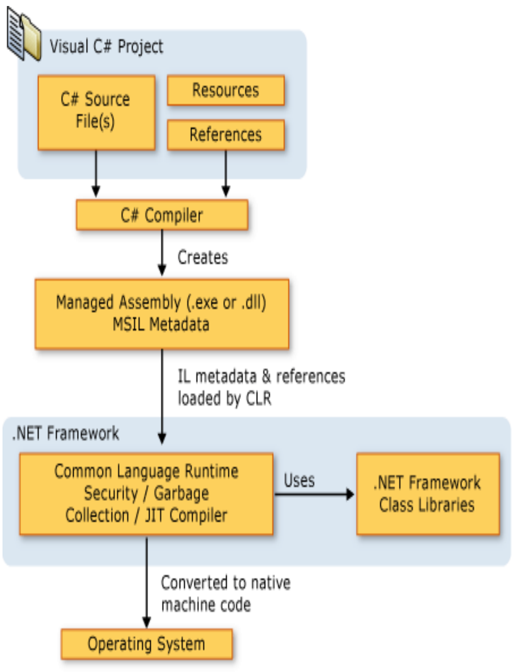

Visual C# Project
C# Source File(s)
Resources
References
C# Compiler
Creates
Managed Assembly (.exe or .dll) MSIL Metadata
IL metadata & references loaded by CLR
.NET Framework
Common Language Runtime Security / Garbage Collection / JIT Compiler
Uses
.NET Framework Class Libraries
Converted to native machine code
Operating System

Fig 1: the .Net data flow [11]

2- The winpcap library, a standard packets sniffing tool [7][17][31], look to Figure 2 :

The winpcap is a packet sniffing library; the main strength of the winpcap is the fact that it bypasses the network protocol stack [7][17][31], this means that this framework will be able to perform activities that might not be allowed by the network interface card.

For example, the winpcap library is able to sniff packets that are received by the Network Interface Card (NIC) but have been dropped probably because the PC was not the packet's destination. That is called sniffing in *promiscuous* mode. Another example would be the packet construction capability that allows for any data to be injected into the packet, for example, a network interface card would drop any packet exiting it that has a different source IP/MAC address than its own, winpcap can bypass this and spoof (i.e. spoofing is creation of a packet with a false source IP/MAC address) a source IP or MAC or any other field. The winpcap creates a driver that provides low level access to the network.

The best feature about winpcap though, is the speed at which the packet is delivered to the higher level API without being too resource intensive. The reason behind this is the packet filter component of the NPF, since thousands or even millions of packets could be coming from/going to the network device, a good filtering technique is required; the filtering method for the winpcap is similar to Unix's BPF (Berkley packet filter) which permits link-layer packets to be sent and received (thus bypassing the protocol stack).
The winpcap takes the filter created at user level and converts it into a low level filter that is then injected into the kernel (i.e. the kernel is a bridge

between the O/S and the hardware) [17]. The filter is not interpreted, but called at a low-level.

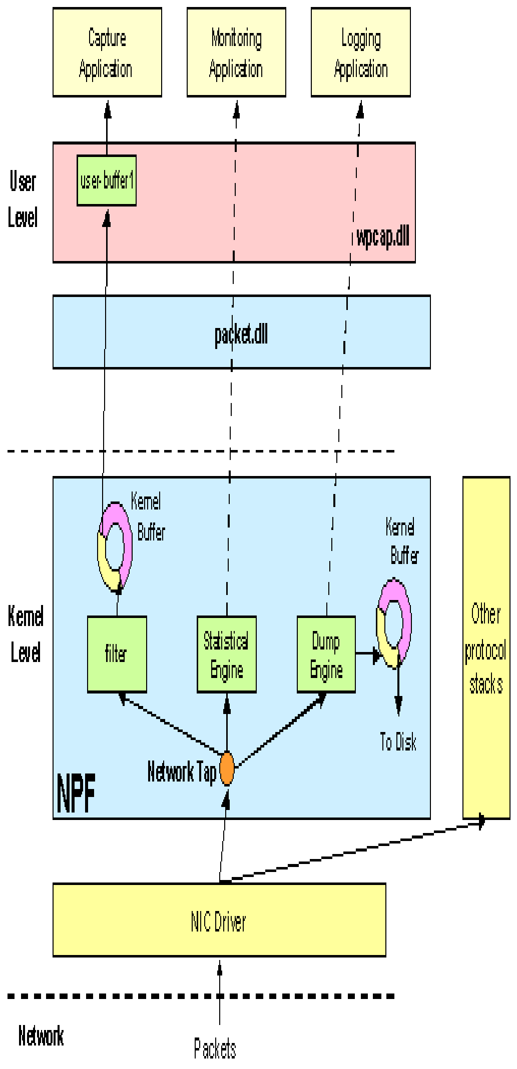


Fig 2: Winpcap's operation, [17]

3- A wrapper for the winpcap library for the C# language (i.e. .Netpcap [22]) :

The winpcap was designed to work with the C++ language; that caused developers to develop and implement wrappers for the original framework to make that framework available for the .Net platform. There are wrappers for java and other languages as well. We have chosen the .netpcap to use. The .netpcap incorporates all the features of the winpcap except for 2 features, the airpcap and remotepcap features. Airpcap has to do with wireless data being sent on the network. The remotepcap is basically a feature that forwards all packets from one network interface card to a remote network.

The technical/economic Value of the solution could be summarized in:

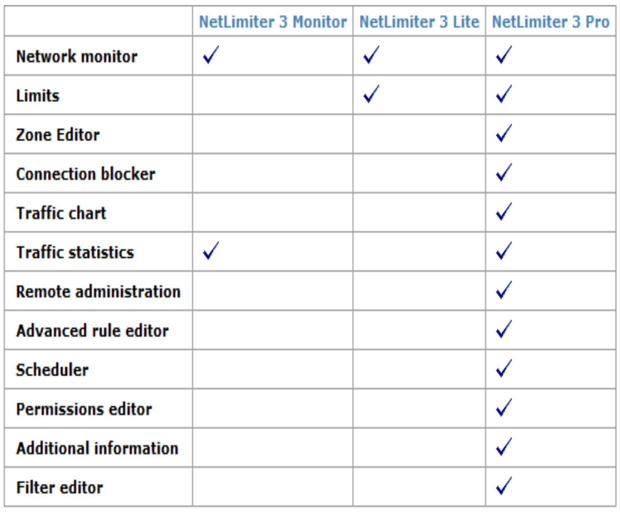
1- The proposed solution is an open source and free of charge. It is a .Net framework that has been developed using MS visual studio's C#. It contains extra features that do not exist in neither commercial nor open source solutions.
2- Upgrading to the commercial (NetLimiter pro.) [14] Network Monitor tool costs $1499.95 for multiple licenses or $30 for a single license. The permanent license of the commercial (OPmanager) [19] Network Monitor tool costs $995 for the professional 50 device pack, $1995 for the essentials 50 device pack and $3995 for the deluxe 50 device pack,
3- Open source Network Monitor tool solution (Wireshark) [6] has been developed using, old framework, ANSI C.

2. MARKET ANALYSIS

2.1 NetLimiter ([14]), look to Table 1

The NetLimiter is an internet traffic monitoring and control tool for the MS windows platform. The NetLimiter incorporates various features such as a network monitor, connection blocker, rule editor and scheduler that mark this tool as a good network monitoring applications. Most of those features are limited however to the full version of the network limiter, the free version only includes the network monitor. Upgrading to the NetLimiter pro costs $1499.95 for multiple licenses or $30 for a single license [14].

Table1. The NetLimiter monitor vs. the NetLimiter pro [15]

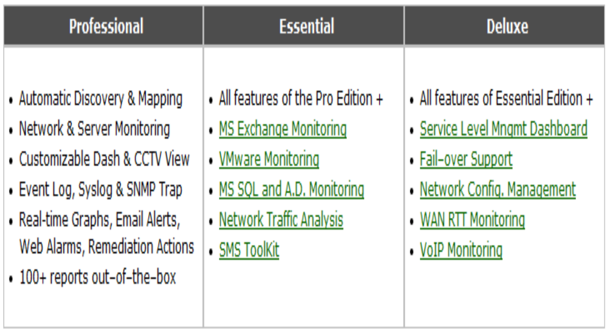


	NetLimiter 3 Monitor	NetLimiter 3 Lite	NetLimiter 3 Pro
Network monitor	✓	✓	✓
Limits		✓	✓
Zone Editor			✓
Connection blocker			✓
Traffic chart			✓
Traffic statistics	✓		✓
Remote administration			✓
Advanced rule editor			✓
Scheduler			✓
Permissions editor			✓
Additional information			✓
Filter editor			✓

2.2 Opmanager([19]), look to Table 2 and Table 3

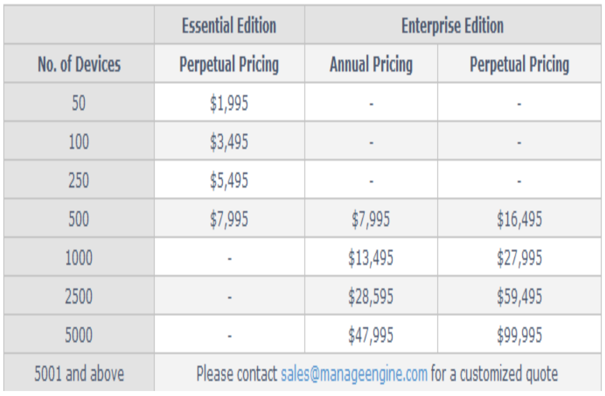
Another network monitoring application, this one targets the corporate world with over 8000 organization using this network tool [19]. The perpetual (permanent) license of the OPmanager [19] costs $995 for the professional 50 device pack, $1995 for the essentials 50 device pack and $3995 for the deluxe 50 device pack. The OPmanager offers many options that distinguish it from other network monitoring tools [19]. Many of those tools are not related to monitoring.

Table 2. OPmanager versions features [19]



Professional	Essential	Deluxe
• Automatic Discovery & Mapping • Network & Server Monitoring • Customizable Dash & CCTV View • Event Log, Syslog & SNMP Trap • Real-time Graphs, Email Alerts, Web Alarms, Remediation Actions • 100+ reports out-of-the-box	• All features of the Pro Edition + • MS Exchange Monitoring • VMware Monitoring • MS SQL and A.D. Monitoring • Network Traffic Analysis • SMS ToolKit	• All features of Essential Edition + • Service Level Mngmt Dashboard • Fail-over Support • Network Config. Management • WAN RTT Monitoring • VoIP Monitoring

Table 3. Detailed pricing for the OPmanager [19]



	Essential Edition	Enterprise Edition	
No. of Devices	Perpetual Pricing	Annual Pricing	Perpetual Pricing
50	$1,995	-	-
100	$3,495	-	-
250	$5,495	-	-
500	$7,995	$7,995	$16,495
1000	-	$13,495	$27,995
2500	-	$28,595	$59,495
5000	-	$47,995	$99,995
5001 and above	Please contact sales@manageengine.com for a customized quote		

2.3 Wireshark([6])

Unlike the other tools, the wireshark is open source software. The wireshark relies on the Winpcap framework for network monitoring, winpcap is a popular packet capturing library [31][17][7]; the wireshark relies on packet capturing to replay and analyze a session. The main feature for the wireshark is its support for many protocols. Wireshark had been operating since 1998 and therefore has built a large database regarding the protocols in use. In addition to that, wireshark has been developed using ANSI C [6] making it a very resource friendly tool, source code is released regularly with each update and can be found at [29].

Limitations of wireshark mainly are:

1- It is a passive network monitor, it simply logs and reports and does not interfere with the network operation whatsoever; this means that it is not possible to limit or kill connections that might be causing any potential damage.
2- It is incapable of real time statistical reporting such as the speed of connections.

Still it is a good and free tool. Limitations could be overcome by adapting the source code to one’s need using, old framework, ANSI C.

3. ANALYSIS AND DESIGN OF NET PORTER

3.1 Development model

The agile development model will be used for this project. The agile development model is an adaptation of the older iterative and incremental model. Common characteristics of both models are that unlike the waterfall model, going back and improve each phase throughout the project is possible [26].

That model will be helpful because:

1. The field of open source software is volatile, it would not be wise to assume that the project will carry on according to plan; additional functionality will be added/removed as the project vision becomes clearer. The waterfall model is more suitable for other non-volatile projects.

2. To increase the speed of project delivery [24]; this is because of the early release of the project and the regular updates that follow, the project will go through the perpetual beta phase for an indefinite period of time. A perpetual beta is a stage where the product enters the Beta phase for an indefinite period of time. The main advantage for the ongoing beta is the fact that the possibility of monitoring the user’s activity [12]. The Net Porter is not a web application and therefore this does not apply here, however, an error reporting functionality logs down any errors to a log file. Normally, at that case, what will be done is the user sending the file along with a description of what could have happened. The main reason behind going into perpetual beta is the unavailability of a testing team. As a result, the user will be the tester and contributor to future changes. The Drawback to perpetual beta is that the project might lose some of its customers if a certain bug drastically affected the application’s operation.

3.2 Aims and Objectives

The core of the application should be the correct display of *who’s doing what*. The processes connected to a remote host should be displayed, along with the IPs being used in those connections, the data sent and received. The required information is mainly numerical or statistical.

Another feature is usability; how long it would take someone who is new to the system to understand how to operate the application is a very important feature, the easier the program is for the user to use and understand the better; in addition to that, the user manual will be the only form of training a non-corporate user will get, so either the application should be self explanatory, or the user manual should make things perfectly clear for the user in a short period of time.

The user will need to store the network session in a file for many reasons (session review, network analysis, archiving…etc), some form of log should be generated that would give the user some idea about what had been going on within the network in that session.

The *minimum* requirements of the application should not be too high bearing in mind that the *minimum* requirements of the application target the average computer user.

Look to Figure 3 for the data flow chart for the main function of the application starting at the event of a packet arrival.

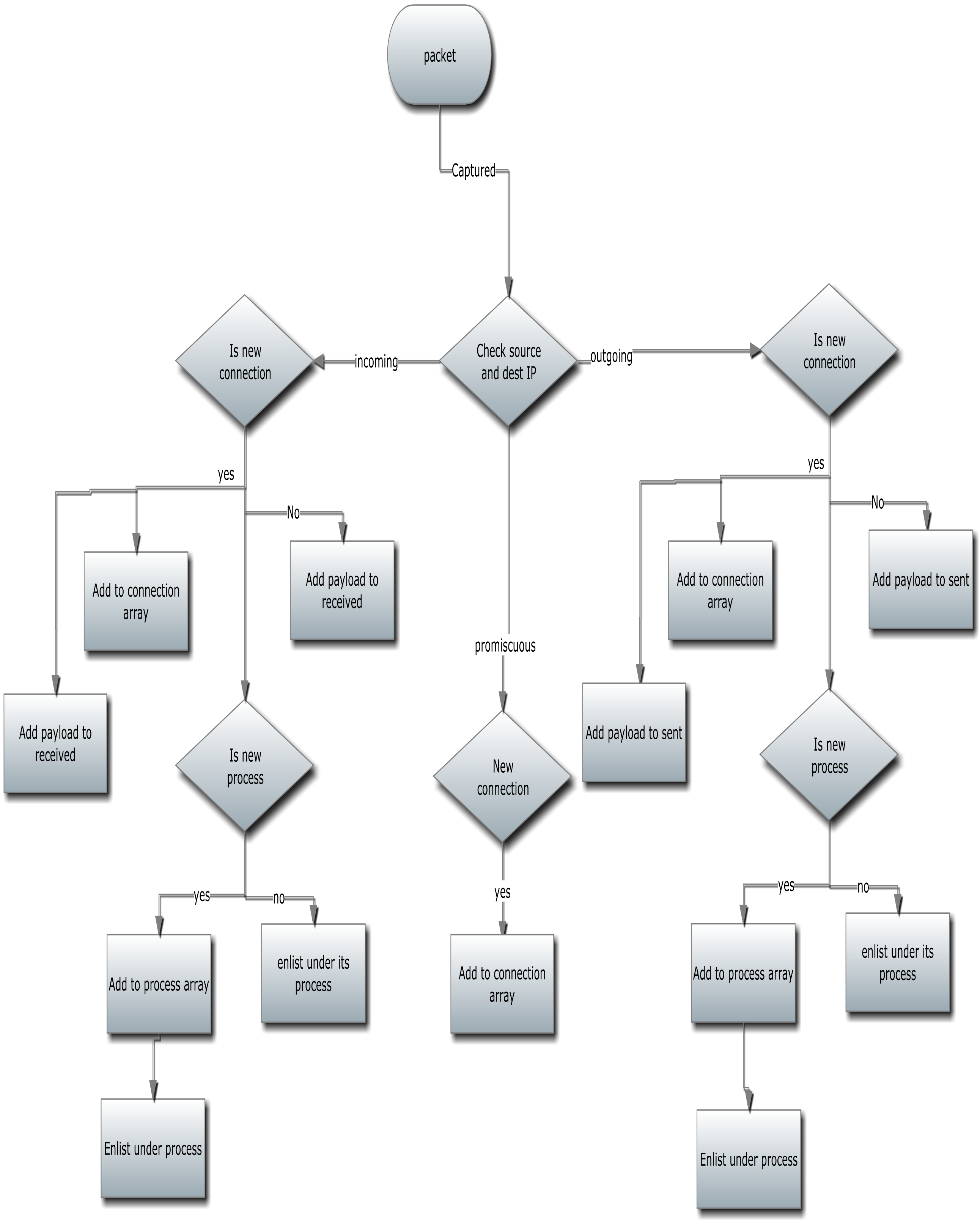


Fig 3: data flow chart.

4. DEVELOPMENT & IMPLEMENTATION OF NET PORTER

The first approach for developing the Net Porter is to use the functions that were available inside the .Net framework. By using raw sockets, it was possible to capture packets at layer 3 of the OSI model (the network layer), look to Figure 4. This means that the data captured came in packet format instead of frame format. That means no Frame Check Sequence (FCS) and no Media Access Control (MAC) addresses.

One problem is that the whole network analysis was extremely difficult since there was no packet object in the .Net framework. Instead what needed to be done was to retrieve the data by specifying the offset and the size.

For example, the item X is the 5th byte and its size is 2 bytes. That requires recognition of the protocol in use during execution time, then create a database that includes how to deal with the recognized protocol in order to determine what type of data can be get and how could be get it, i.e. offsets and fields. That is time consuming and resource inefficient as well. The dotnetpcap [2][22] solves the abovementioned problem. The Net Porter’s *connection* class is an important part of the project, without that class the Net Porter is just another packet dumping application, look to Table 4.

In Table 4, there are value assigners for all data types. The use of those assigner functions is to modify the values of the members safely with minimal problems. An array of connection objects is the backbone of the Net Porter, it holds all the network connections along with their members, and indexes them according to their sequence of arrival. The purpose of this array is to hold and to organize data, and to make it easier to define if the incoming/outgoing connection is already in the list or not. This array is initiated to a value of 1 connection. The constructor of this object simply creates a new object with null values that will be filled in later. The array is a dynamically expanding 1 (we first start with 1 empty connection, as soon as a connection comes in, the array is resized into 2, holding 1 actual connection and 1 empty connection, there is always an empty connection to make room for an incoming 1).

Table 4. a) some members of the connection class, b) some global data types

a)

IP	String	Holds the IP of the remote host
Process	String	The process using/listening on that connection
Details	String	The window handle for the process
Sent	Int	Total data sent by the connection in bytes
Received	Int	Total data received in bytes
Rsamplea, rsampleb	float	2 samples used to determine a temporary speed sample
Rspeeda, rspeedb	float	2 speed samples over the past 2 seconds to determine average speed
Rspeed	float	The estimated download speed of the connection
ssamplea, ssampleb	float	Same as rsample a and b
Sspeeda, sspeedb	float	Same as rspeed a and b
Sspeed	float	The estimated upload speed of the connection

b)

Connections	Array of the object connection	Stores the connections
Newupload, newdownload, newprom, newprocess	Boolean values	Helps define if the connection is already stored in the connections array, and if the process is in the processes array
Totalconn, processno	Integer value	Number of connections/processes in the array, helps in resizing the array and iterating through the array
Timeelapsed	stopwatch	Calculates the time the application has been running
Alldevices	Ilist of livepacketdevice	Holds all the NICs for selections
Deviceindex	Integer value	References the NIC to be used

Tdr, tds	Integer values	Hold the amount of data sent/received in bytes
Tdr_display, tds_display	Integer values	Hold the amount of data sent/received in the user's desired unit (KBs/bytes)
Downspeeddisplay, upspeeddisplay	Integer values	Hold the download and upload speed
Downsample, usample	Array of 2 float values	Hold the upload/download samples that are used in calculating the speed
Samcondition	Integer value	0,1or 2 then resets, to take samples of the data rate
Last, last2	Integer values	Hold the last samples for temporary data rate calculation
Downspeeed,upspeed	Integer values	2 more samplers for the data rate
Communicator	PacketCommunicator	The device in use
dumpfile	PacketDumpFile	The dump file for logging traffic(if any)
tn	TreeNode	Temporary storage value for tree nodes to be added
Selectedindex, comparison index	Integer values	Used to determine if the selected tree node had been changed
Settings	Array of strings	Stores the settings loaded from the settings.dat
Processes	Array of strings	Like the connection array but stores running processes

4.1 The packet handler

To describe what happens when a frame arrives within the network, we must first know what happens before data is sent. In 1984 the International standards organization (ISO) developed the Open System Interconnection (OSI) model as a reference model for network communication [9]. So here's the scenario, application x wants to send a piece of data over the network, a header is appended at each layer of the OSI, look to Figure 4, the chunk of information to be sent has a different name at each layer starting the transport layer down (or up), look to Figure 5. The process is reversed at the receiving end and headers and trailers are de-capsulated at each layer until the data is back in its original form at the application layer [4][8][23].

The winpcap operates at layer 2 of the protocol stack. This means that the unit of operation is the frame. When a frame is captured by the kernel filter application, the frame can either match the filter criteria or not. The packethandler(packet) function handles the incoming packets. For simplicity, the unit will be referred to as packet from here on even though it is a frame for standard purposes. If dumping is enabled, the packet is appended to the traffic log file.

The very first operation is to determine what type of traffic (incoming, outgoing or other) using the source and destination IP. If the destination IP matches the User's IP then it is considered as incoming traffic and vice versa. We have to check if the remote IP, source for incoming and destination for outgoing, already exists in the connections array, in case it does, the packet payload, amount of data excluding headers, is added to the data sent/received member of the connection. If it does not exist in the connections array, a new connection object is created and the source/destination IP is assigned to

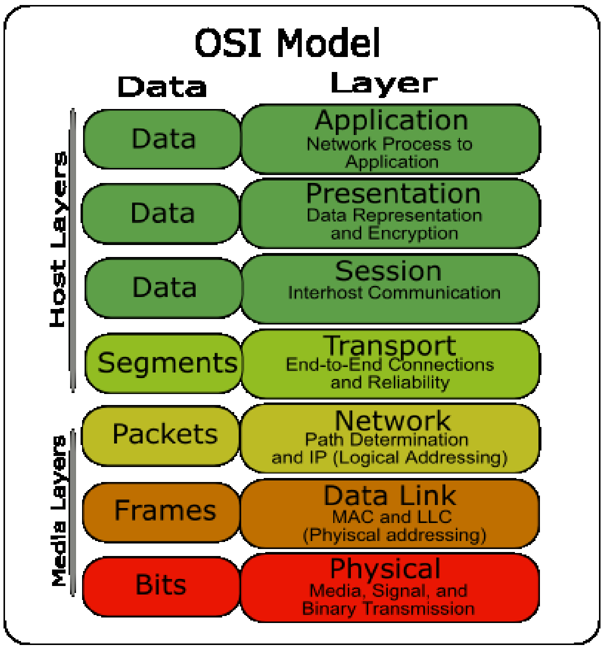

OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Phyiscal addressing)
Bits
Physical
Media, Signal, and Binary Transmission

Fig 4: the OSI model [20]

the IP attribute, the array's size is increased by 1 and the total connection counter is incremented.

Then, the process that is using the connection needs to be determined. This is where the interoperability functions of C# become present and the Pinvoke (i.e. platform invoke) is used. The purpose of a platform invoke is the use of native functions/dlls on managed code, or to make it simpler, to call native win32 functions inside the C# development environment [4][23][8].

The getextendedtcptable() function is used to associate the connection with a process. Afterwards, that process' name is stored in the connection's process member and the process handle. The text on the taskbar of the process is stored in the details member. Then that process is check if it exists within the processes array, if it does, the IP is added as a child for the tree node representing the process, if not a new tree node is created for that process, and the IP is added as a child for that tree node, and the process name is added in the array of process and similar to what had been done before, the array's size is expanded by 1 to accommodate the next process to come.

4.2 Promiscuous mode

Any data that does not match the sorting criteria, has neither the source nor destination IP the same as the device's IP, is labeled as promiscuous data. This might include broadcasts for example; a broadcast is message to all interface cards on a network [16], this might include a routing update, and an Address Resolution Protocol (ARP) request, used to link Mac addresses with IP addresses via broadcasting an ARP request.

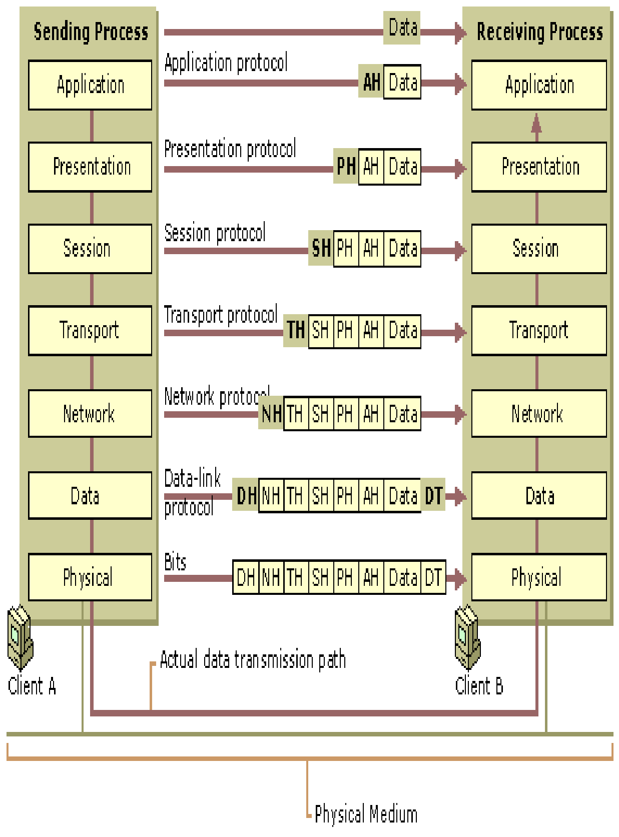


Fig 5: data flow in the OSI model [4]

Promiscuous data is more interesting in a hub environment, since hubs forward frames on all its available ports, all data traversing throughout the network can be sniffed and logged. The promiscuous data process is always first in the list and has an associated tree node as soon as the program starts. Any data not classified as incoming or outgoing traffic is placed under the promiscuous data category and treated as such. Promiscuous data is data that does not belong to the user, therefore it does not account for the total data sent/received that, the winpcap library is incapable of retrieving the payload of promiscuous data, so it does not include the total data part as well; it is only included as reference for the statistical and traffic log (included in both). Promiscuous data connections are stored in the array just like any other connection with the exception that the stored IP is stored in the following format:

<Source IP> "-->" <destination IP>

(192.168.1.1-->255.255.255.255 for example).

4.3 System requirements for Implementation

4.3.1 *Minimum Requirements*

- Intel Pentium III clocked at 1.0GHZ(or equivalent)
- 20 MBs of free memory
- Winpcap runtime library downloadable (http://www.winpcap.org/install/default.htm)
- .Net framework 4.0 downloadable (http://msdn.microsoft.com/en-us/netframework/aa569263.aspx)
- Windows XP SP2 or better.
- A Network interface card compatible with the winpcap runtime

4.3.2 *Recommended Requirements*

- Intel Pentium IV clocked at 2.0GHZ(or equivalent)
- 40 MBs of free memory
- 256 MBs of storage space.
- Winpcap runtime library downloadable (http://www.winpcap.org/install/default.htm)
- .Net framework 4.0 downloadable (http://msdn.microsoft.com/en-us/netframework/aa569263.aspx)
- Windows XP SP2 or better.
- A Network interface card compatible with the winpcap runtime

5. TESTING OF NET PORTER

As soon as the Net Porter application is opened, an interface that appeared at Figure 6 will be appeared. Software testing is the process used to measure the quality of developed computer software [28]. Testing is a necessary process in the lifetime of any project. It serves as a measure for the completeness, readiness and correctness of the software as well as to assess the project's relevancy to the specification documents and the following of standards and regulations.

5.1 Regression test

Regression test is the retesting of any unchanged parts within the application when an update or a change takes place. This is mainly to verify whether the new change or feature had introduced any bugs [27]. This type of test is generally useful when most controls or functions are related or operate on the same members of the application. For example, the array of the connection object is used by almost everything in the application, the start/stop button for example, clears the whole array, while the packet function adds new connection into that array, and the statistical log function reads the whole array. An error in the main function would definitely affect anything in the application. This does not mean that every single function will be retested with every new version of the program. That will be time consuming. Instead a map will be constructed to provide information about which functions access which overlap in terms of access, look to Figure 7. Another critical reason would be the fact that the application is multithreaded any update to a function might affect the functions done by other threads [21].

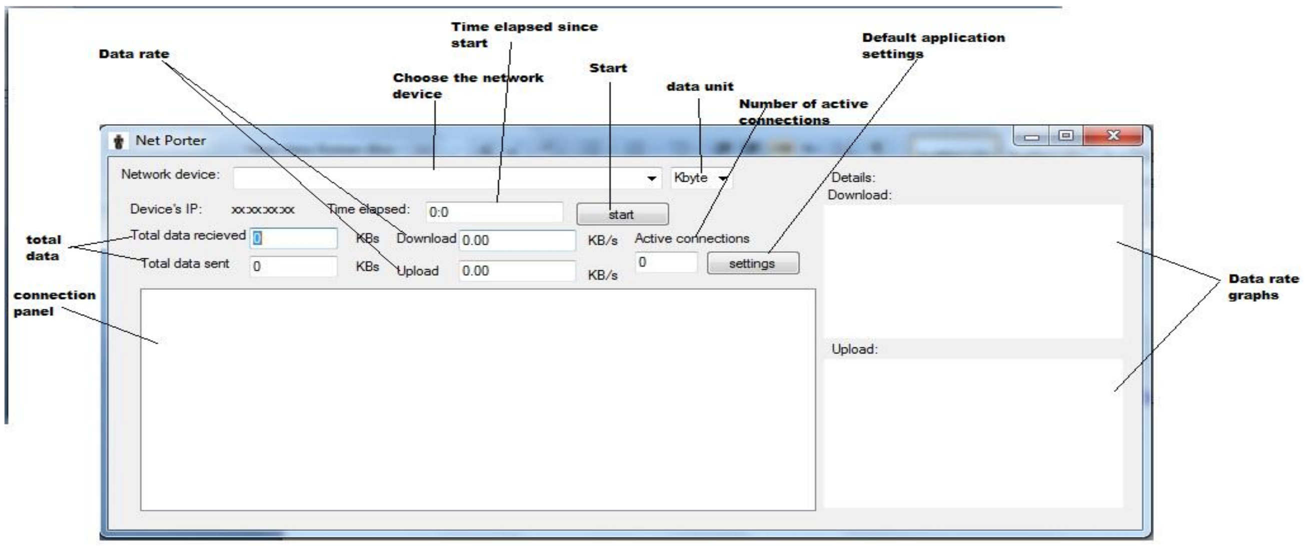


Fig 6: the Net porter's main Interface

By referring to the regression test map, look to Figure 7, it can easily be deducted which functions need to be tested when a function has been updated. The relation is simple, if any function that writes to an object is updated, all functions writing to and reading from that object are updated as well; for instance, updating the processcalc_tick function means that testing all functions branching from the TN object but not all items branching from the connection array have to be done. Changing the statslog_tick would not need any retesting because it does not change any item within the application.

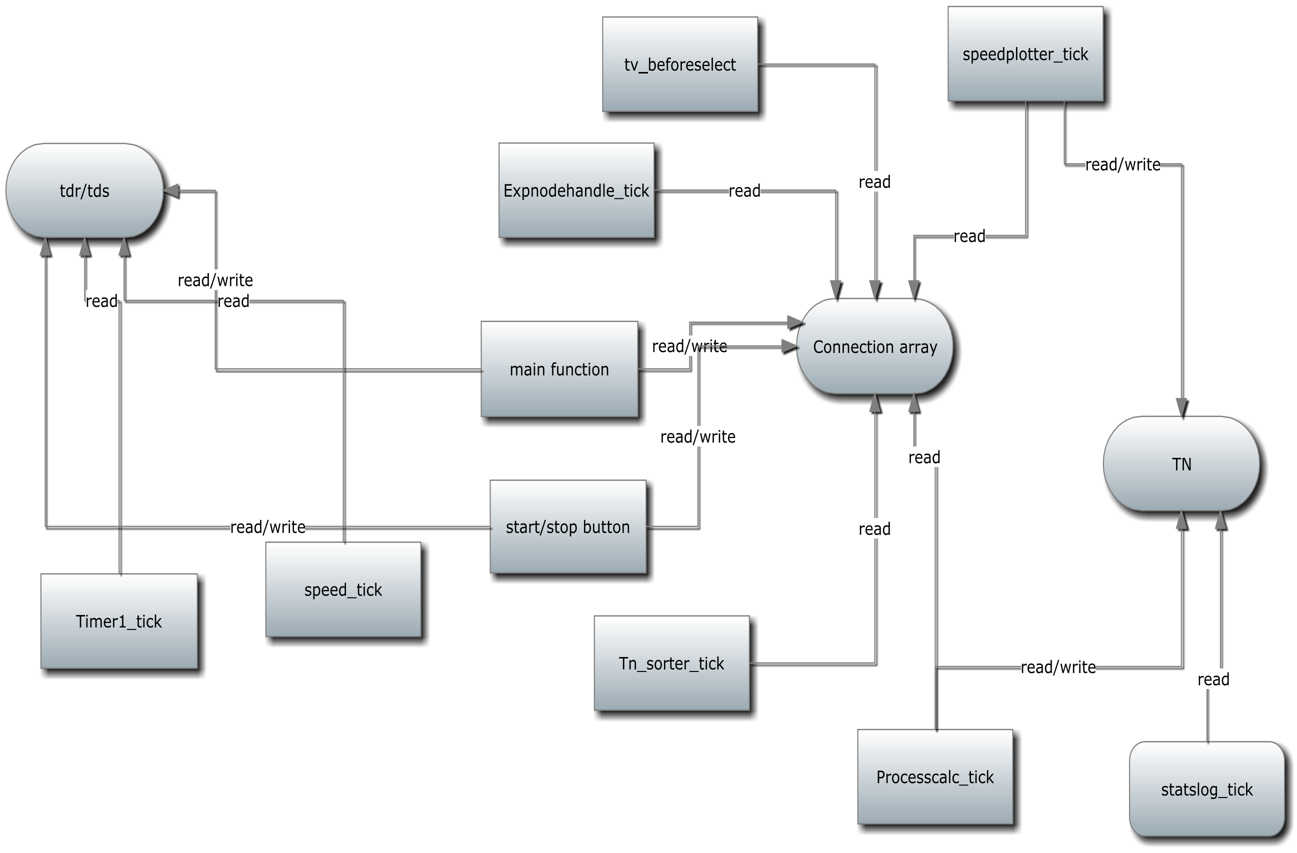


Fig 7: the regression testing map

5.2 Positive /negative test cases

Positive testing is attesting methodology that attempts to show that a certain module does what it is supposed to do. Negative testing attempts to show that a certain module does not do what it is not supposed to do. Both are concepts rather than activities or standards [18].

The concept itself will be applied to certain degree. Testing is difficult in the Net Porter because of not having absolute control over the application's input. That implies that testing is limited to items that can be controlled while the incoming data will be assumed to be similar enough for test cases or test scenarios.

Initially, the resource allocation for the testing was as follows:

1.	Preliminary testing	03 days
2.	White box testing	14 days
3.	Black box testing	14 days

This plan was not carried out due to the lack of testers that makes it difficult to conduct the black box testing. In addition to that the adoption of the agile method increased the amount of preliminary testing/informal testing since it is needed to test any function before it was released, and thus the preliminary testing did not have a dedicated time slot but stretched over the development period. The preliminary testing was estimated to have taken 7 days. White box had to be extended to attempt the covering of the missing black box testing process, and was extended to 22 days. Look to Table 5 for some test cases.

Table 5. The following ID=01 up to ID=06 are test cases. Test case template retrieved from [10].

ID=01

Unit to test	**Test data**	**Steps to be executed**	**Expected result**	**Actual result**	**Pass/fail**	**Comments**
support for Wi-Fi. **Assumptions (none)**	Intel® Wi-Fi link 5100 agn wireless adapter	1-Turn off all network devices. 2-Connect using the Intel Wi-Fi adapter. 3-Browse a random webpage. 4-Wait for a local network broadcast	Normal traffic should be logged, broadcasts should be added under promiscuou s data.	Regular traffic was logged and properly categorized, broadcasts were added under promiscuou s data.	pass	More Wi-Fi interface cards need to be tested, the layer 2 protocol in use was Ethernet II.

ID=02

Unit to test	**Test data**	**Steps to be executed**	**Expected result**	**Actual result**	**Pass/fail**	**Comments**
support for Optical fiber protocols. **Assumptions (Network traffic will not affect this test case)**	a fiber optic enabled protocol	1-Turn off all network devices. 2-Connect using the FC network card. 3-Browse a random webpage. 4-Wait for a local network broadcast.	Normal traffic should be logged, broadcasts should be added under promiscuou s data.	Could not be conducted, no FC capable network card was available	fail	Will be added to the future agenda

ID=03

Unit to test	**Test data**	**Steps to be executed**	**Expected result**	**Actual result**	**Pass/fail**	**Comments**
data calculation in bytes. **Assumptions (none)**	random traffic	1-Start monitoring traffic. 2-Browse a random website. 3-Stop all network activity. 4-Allow the statistical log to run once for KBs unit and once for byte unit. 5-Compare the result.	The unit conversion should yield a correct result.	The conversion's results were correct	pass	None

ID=04

Unit to test	**Test data**	**Steps to be executed**	**Expected result**	**Actual result**	**Pass/fail**	**Comments**
Graphs. **Assumptions (network is stable, minimal jitter)**	random traffic	1-Start monitoring traffic. 2-Initiate a download that can last throughout the test case. 3-select an IP to start plotting the graph. 4-select another IP. 5-select a process.	The graph plotting should begin after an IP is selected, upon selection of another IP, the graph should automatically clear and stat plotting the new IP, the process should plot its traffic as well.	Graph plotting was fine, when the selection was changed, the graph was cleared and started plotting, upon the selection of a process though, the graph plotter froze	conditional pass	The process plotter will be disabled until it is correctly working, will be added to future agenda

ID=05

Unit to test	Test data	Steps to be executed	Expected result	Actual result	Pass/fail	Comments
terminating connections and/or processes. **Assumptions (The monitoring process will not go wrong)**	random traffic	1-Start monitoring traffic. 2-Initiate a download. 3-Kill the download for the IP of that download. 4-Initiate an upload. 5-Kill the upload for that session. 6-Initiate another download. 7-Kill all connections for the process downloading. 8-Kill the process.	Kill download/upload should terminate the upload/download. Kill connection(process) should kill all connections within the process, upon selecting kill process the process should terminate.	All kill functions worked fine.	pass	None

ID=06

Unit to test	Test data	Steps to be executed	Expected result	Actual result	Pass/fail	Comments
performance under heavy load. **Assumptions (no external variables will affect the performance)**	many random connections, a core 2 duo machine (2.00 ghz)	1-Start monitoring traffic. 2-Initiate as many connections as possible. 3-Use the task manager to monitor performance.	CPU utilization should not exceed 50%.	The CPU utilization maxed out at 43%	pass	Some performance improvements will be added to counter such scenario (or attack).

6. CONCLUSION AND FUTURE WORK

Look to Table 6 for a comparison between Net Porter, Net Limiter and Wireshark.

There are two types of network monitoring, active and passive. As implied by the name, the passive network monitoring simply monitors and/or logs information [3]. The main difference between active and passive monitoring is that active monitoring interferes with the network connections proactively, reactively or at user command [3]. The difference between the two types is evident in an intrusion detection system (passive) and an intrusion prevention system (active).

It would definitely add value to the program if it were capable of interfering with the connection as the user might need; therefore, killing the connection and the throttling (limiting) of bandwidth will be set as secondary objectives at this point.

Write for an audience, code tends to outlive jobs. The coding style and standard makes a difference in this situation; a more critical difference than any commercial product; this is an open-source application, and the source code will be released for others to adapt to their own needs, as a result, proper coding standards should be followed and the code should be clear and understandable for any editor. This can be achieved by following common developer community practices and standards, use proper code indentation and coloration, and simply by adding proper comments when necessary [25].

Furthermore, the stability of the program constitutes a major aspect of a good network monitoring system. This is why the stability of the system should be given attention. For the Net Porter, the top priority under the stability title will be *loss of data*. In case of an application crash or a system failure the data logged throughout the session should be intact. Add to that minimizing of system crashes and system failures, and of course a proper alarming mechanism to alert the user in case of a system failure.

The final feature is to ensure the confidentiality of the information generated by the application; this is the least significant feature. The user can use an external program to encrypt/decrypt the data, yet having the function in the same application will definitely score some points in the user's appeal for the application.

Throughout the development, implementation and testing phase, the upcoming agenda of the project has been arising. That Agenda, in ascending order of priority is:

1. **Process plotter function:**

Not a major function but has major uses. It is mainly aesthetical and for usability uses. The whole point is to create an object named processes and creating an array of that object, similar to the connection array, and the object would have its own members like total data sent and received to allow for individual sampling for the speed and to allow graph plotting of a process's state, like the IPs.

2. **Level 4 protocol information:**

There are two levels of connection in the connections tree, the second level is an IP. What is planned is adding a third level which is ports, adding ports will not only offer more flexibility for the user, in term of killing a port's connection, but will also add more accurate support for processes because in the current implementation, if two processes are using the same IP, the first process in the TCP Table entry determines the process that will enlist that IP as a child.

3. **More sophisticated filter:**

The layout is already set to accept a filter from a string input, it is planned however to make filter creation simpler for the user; this could be done by adding a set of selection, using menus, drop down boxes, auto completion...etc, that will make the user's life much easier.

4. **Client-server version:**

Adding a client-server versions of the application will be very useful in large or medium sized networks, a server could have the ability to kill connections, retrieve logs or change the settings for example, while a client server complies. This will be very useful for remote monitoring in large companies in particular.

5. **Encryption/decryption of logs:**

As previously mentioned, this feature is not of great significance, but it would be more pleasant of the user can encrypt and decrypt within the same creator application, it's more of a luxury than a necessity.

Table 6. A comparison between Net Porter, NetLimiter and Wireshark [1]

	Net porter	**NetLimiter**	**Wireshark**
Network monitoring	x	x	x
Traffic logging	x		x
Network statistics	x	x	
Processes/applications	x	x	
Statistical logging	x		x
Bandwidth throttling		x	
Runs in background	x	x	
overall data statistics	x		
Automated logging	x		
Performance	Very Good	Excellent	Good
Killing process/connection	x	Only Connections	
Graphs	x	x	

7. REFERENCES

[1] A. A. A. H. Adel, June 2012, The Net Porter A Network Monitoring Application The .Net Approach, Graduation Project, Supervised by Dr. Islam Taj-Eddin, ICS, BUE.

[2] About CodePlex, CodePlex Information and Discussion, CodePlex Project Hosting for Open Source Software.http://codeplex.codeplex.com/

[3] Cottrell Les, March 11th 2011, Passive vs. Active Monitoring, Stanford Linear Accelerator centre (SLAC).http://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html

[4] Data Flow in the OSI Model, 2012, OSI Model, Appendixes, TechNet, Microsoft. http://technet.microsoft.com/en-us/library/cc977591.aspx

[5] Define *porter*, Dictionary.com.http://dictionary.reference.com/browse/porter?s=t

[6] Development and maintenance of wireshark, 2012, wireshark Developer's Guide for wireshark 1.9, wireshark' Documentation, wireshark the world's foremost network protocol analyzer. http://www.wireshark.org/docs/wsdg_html_chunked/ChIntroDevelopment.html

[7] Frequently Asked Questions, October 19th 2009, WinPcap The industry-standard packet capture library.http://www.winpcap.org/misc/faq.htm#Q-25

[8] Gold Mike, October 6th 2005, Introduction to Multithreading in C#, Multithreading in C#, C# Corner.http://www.c-sharpcorner.com/uploadfile/mgold/multithreadingintro10062005000439am/multithreadingintro.aspx

[9] Hiemstra Johan, 2011, 7-Layer OSI Model, CCNA techNote, Techexams.net.http://www.techexams.net/technotes/ccna/osimodel.shtml

[10] How to Write Effective Test Cases, 2007, Software Testing Help.http://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

[11] Introduction to the C# Language and the .NET Framework, 2012, Getting Started with Visual C#, Visual C#, MSDN Library, Microsoft.http://msdn.microsoft.com/library/z1zx9t92

[12] Morris Jim, August 30th 2006, Software Product Management and the Endless Beta, Carnegie Mellon University.http://jimmorris.blogspot.com/2006_08_01_jimmorris_archive.html

[13] Nash S. Kim, Behr Alyson, June 5th 2009, Network Monitoring Definition and Solutions, CIO Magazine.http://www.cio.com/article/133700/Network_Monitoring_Definition_and_Solutions?page=1&taxonomyId=3071

[14] NetLimiter-ultimate bandwidth shaper, 2012.www.netlimiter.com/

[15] NetLimiter 3 feature list and comparison table, 2012, Netlimiter.com.http://www.netlimiter.com/featurelistnl3.php

[16] Network Broadcasting and Multicasting, Networking Tutorial, Networking, Independent Technologies The Computer Technology Documentation Projec http://www.comptechdoc.org/independent/networking/guide/netbroadcasting.html

[17] NPF driver internals manual (WinPcap internals), 2008, WinPcap Documentation (4.1 beta5), WinPcap The industry-standard packet capture library.http://www.winpcap.org/docs/docs_41b5/html/group__NPF.html

[18] Nyman Jeff, 2002, Methodology and techniques, SQAtester.com.http://www.sqatester.com/methodology/PositiveandNegativeTesting.htmhttp://www.globaltester.com/

[19] OpManager, 2012, ManageEngine.ca, Optics Engineering.http://www.manageengine.ca/opmanager.aspx

[20] OSI Model, April 5th 2010.http://diana1110.blogspot.com/2010/04/osi-model.html

[21] Pattison Ted, September 2004, Basic Instincts: Thread Synchronization, Issues and Downloads, MSDN Magazine.http://msdn.microsoft.com/en-us/magazine/cc163929.aspx

[22] Pcap .Net, CodePlex Project Hosting for Open Source Software.http://pcapdotnet.codeplex.com/

[23] Platform Invoke Tutorial, 2012, C# Tutorials, C# Programmer's Reference, MSDN Library, Microsoft.http://msdn.microsoft.com/en-us/library/aa288468(v=vs.71).aspx

[24] Priest Matt, May 7th 2012, Agile Development Models for Consumer Goods – Not Just for Software Anymore, Kalypso viewpoints, Kalypso Delivering on the promise of Innovation.http://kalypso.com/viewpoints/resource/agile-development-models-for-consumer-goods-not-just-for-software-anymore/

[25] Pritzker Yan, September 29th 2009, Five Rules for Writing Good Code.http://yanpritzker.com/2009/09/29/five-rules-for-writing-good-code/

[26] Ramesh Ritesh, February 9th 2012, Agile development, General, Game Development, General Programming, The Code Project.http://www.codeproject.com/Articles/325832/Agile-development

[27] Regression Testing with Regression Testing Tools and Best Practices, 2007, Software Testing Help.http://www.softwaretestinghelp.com/regression-testing-tools-and-methods/

[28] Software Testing Process, January 28th 2009, Topics, Toolbox.com.http://it.toolbox.com/wiki/index.php/Software_Testing_Process

[29] The source code repository, 2012, wireshark.http://anonsvn.wireshark.org/viewvc/

[30] VS Rajesh, October 25th 2001, Pointers in C#, C# Language, C# Corner.http://www.csharpcorner.com/UploadFile/rajeshvs/PointersInCSharp11112005051624AM/PointersInCSharp.aspx

[31] WinPcap The industry-standard packet capture library, 2012.http://www.winpcap.org/default.htm