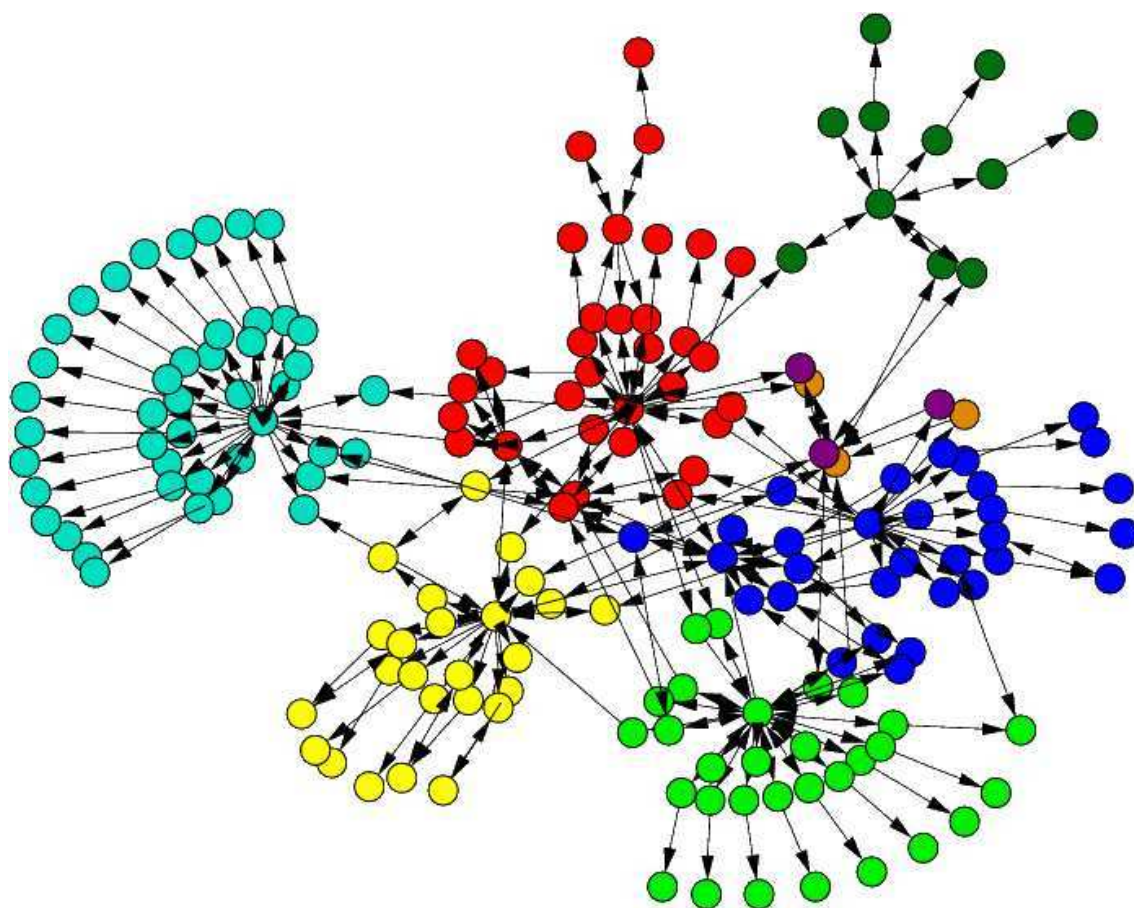


Passive Distributed Network Analysis Using Remote Packet Capture In Java

Thomas C.A. Judge
Year of Study: 2004-5
Supervisor: Daniel Spooner

June 18, 2005



Copyright ©2005 Thomas C.A. Judge

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the following Invariant Sections: "Introduction", "Assessment of the project", "Notes from SourceFire Seminar", "Conclusion", one Front-Cover Text: "Passive Distributed Network Analysis Using Remote Packet Capture In Java", no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Abstract

Intrusively monitoring the activity on a network can add extremely large load to a server, for example monitoring the web sites that users on a LAN visit without having the overhead of running a transparent proxy server. Another example would be to monitor the conversations that people on a LAN are having with the outside world via some form of instant messaging application (i.e. MSN Messenger). Both of these processes would traditionally require a proxy server to intercept the content of messages/pages between the source and the destination, adding extra overhead to systems that could be utilised elsewhere in the organisation. By monitoring this content in a passive fashion it is possible to monitor systems without touching or reconfiguring them.

Keywords

Network, Packet Capture, IP, TCP, UDP, Analysis, Distributed

Preface

During this document the symbol in Figure 1 is used to define a predefined process. This process may be documented as another flow chart at another point in this document. There are also a number of key terms and abbreviations that are only referenced here due to the vast number of times that they appear in the document.

- IEEE 802.3 CSMA/CD Ethernet (802.3)[1]
- Internet Protocol (IP/IPv4)[3][4]
- Address Resolution Protocol on IEEE 802.3 Networks (ARP)[4]
- Reverse Address Resolution Protocol (RARP)[5]
- User Datagram Protocol (UDP)[6]
- Transmission Control Protocol (TCP)[7]
- Network Intruder Detection System (IDS)
- Network Intruder Prevention System (IPS)
- Java[18]
- Java Native Interfaces (JNI)
- Java Virtual Machine (JVM)
- Libpcap - Network Packet Capture API[9]
- Jpcap - JNI Wrapper For Libpcap
- Libnids - Network Packet Capture API[10]
- Open Source Message Queue (OSMQ)[15]

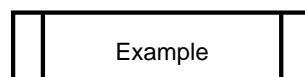


Figure 1: Predefined Process

Contents

1	Assessment of the project	8
1.1	What is the technical contribution of the project?	8
1.2	Why should this contribution be considered either relevant or important to computer systems engineering?	8
1.3	How can others make use of the work in this project?	8
1.4	Why should this project be considered an achievement?	8
1.5	What are the weaknesses of this project	8
2	Introduction	10
2.1	The Problem	10
2.2	The Solution	11
2.3	Project Management	12
2.3.1	Technologies Used	12
2.4	Testing	13
2.4.1	Final Timetable	13
3	The Node	15
3.1	Analysis Engine	15
3.1.1	Capture Engine	16
3.1.2	Packet Processing Engine	17
3.1.3	Network View	18
3.1.4	Stream Assembler	20
3.1.5	Defragmentation Engine	22
3.1.6	Module Dispatcher	23
3.1.7	OS Fingerprinting Engine	24
3.1.8	Traffic Analysis Module	25
3.1.9	MSN Messenger Analysis Module	26
3.2	Hub Communications	27
3.2.1	Status Message	27
3.2.2	Command Message	28
3.2.3	Data Message	28

3.2.4	Acknowledgement Message	28
3.2.5	Node Communication Startup Process	29
3.3	Module Loader and Logging	30
4	The Hub	31
4.1	Node Communications	31
4.2	Persistent Storage	32
4.3	GUI Communications	32
5	The GUI	33
5.1	Network Management Dashboard	33
5.2	Module Management Dashboard	33
5.3	Network View Dashboard	34
5.4	Traffic Analysis Module Dashboard	34
5.5	MSN Messenger Module Dashboard	34
6	Conclusion	35
6.1	Achieved Objectives	35
6.2	Difficulties Encountered	37
A	Appendix: Ethernet Frame Header	39
B	Appendix: IP Packet Header	40
C	Appendix: TCP Packet Header	41
D	Appendix: UDP Packet Header	42
E	Appendix: Notes from SourceFire Seminar	43
E.1	Current Implementation Failings	43
E.2	Features requested by prospective customers	43
F	Appendix: Bibliography	44
G	Appendix: On The CD	46
G.1	Building from source	47

H Appendix: GNU Free Documentation License	48
1. APPLICABILITY AND DEFINITIONS	48
2. VERBATIM COPYING	50
3. COPYING IN QUANTITY	50
4. MODIFICATIONS	50
5. COMBINING DOCUMENTS	52
6. COLLECTIONS OF DOCUMENTS	52
7. AGGREGATION WITH INDEPENDENT WORKS	53
8. TRANSLATION	53
9. TERMINATION	53
10. FUTURE REVISIONS OF THIS LICENSE	53
ADDENDUM: How to use this License for your documents	54
I Appendix: The GNU General Public License	55

List of Figures

1	Predefined Process	3
2	System Over View	11
3	Frame Capture Engine	16
4	Packet Processing Engine	17
5	Example Network	18
6	Simple Network	19
7	Realistic Network View	19
8	Current Stream Assembler	20
9	Original Stream Assembler	21
10	Packet Defragmentation Engine	22
11	Module Dispatcher	23
12	Screenshot showing traffic analysis plugin output	25
13	Hub Connection Overview	27
14	Node Communication Startup Process	29
15	Hub Architecture Overview	31
16	Screenshot showing network management dashboard	34
17	Screenshot showing module management dashboard	35
18	Screenshot showing network view dashboard	36
19	Screenshot showing traffic analysis plugin output	37
20	Ethernet Frame Header	39
21	IP Packet Header	40
22	TCP Packet Header	41
23	UDP Packet Header	42

List of Tables

1	Network Information Records Created From Received Packets	18
---	---------------------------------------------------------------------	----

1 Assessment of the project

1.1 What is the technical contribution of the project?

Packet capture and analysis programs are very complex pieces of software that require large amounts of compute power to achieve their goal, to this end most applications are written in C for greater efficiency. By distributing the system it allows for lower power cheaper systems to be used to monitor a high bandwidth connection. Also writing the system in Java makes it easy to migrate the system to any host operating system supported by Libpcap and Java.

1.2 Why should this contribution be considered either relevant or important to computer systems engineering?

Computer networks are rapidly expanding to cover most of the world, and Computer Systems Engineers are being contracted to design and install them. In order to fully audit the security of any network it needs to be monitored for threats, or to gather performance related information. JxNAP will allow network designers to fully monitor network installations and capture security and performance related information tailored to their exact needs.

1.3 How can others make use of the work in this project?

All of the published documents for this project will eventually be published on the project website (<http://jxnap.tomjudge.com>) under the GNU Free Documentation License¹ under which this document is licensed. All of the source code for the project is licensed under the GNU Public License² and will also be published on the project website. Once all of this is published the project will be registered in the OSTG software index online at <http://freshmeat.net> in order to make the largest number of people aware of the project as possible.

1.4 Why should this project be considered an achievement?

The project makes distributed network analysis software available to everyone, where before it was limited to a selection of commercial products, and only to those that could afford to buy licenses for the products. There is a functioning prototype that demonstrates the capabilities of the system.

1.5 What are the weaknesses of this project?

The main weakness of this project is the lack of target based IP defragmentation and TCP stream reassembly, which allow standard IDS/IPS evasion techniques to be used to allow attackers to traverse

¹Please see Appendix H for a complete copy of the GNU FDL

²Please see Appendix I for a complete copy of the GPL

the system without the threat they present being detected. Another weakness is the OSMQ system that is used for node to hub communications, the due to lack of documentation it has not been possible to fully stabilise the communications between the node and the hub. The any subsequent releases of JxNAP should find an alternative message brokering system.

2 Introduction

2.1 The Problem

In today's complex corporate IT world security is becoming more and more important as the number of systems that are exposed to uncontrolled network traffic is rising. With the hard edge of the business network rapidly diminishing the job of patrolling the network edges is becoming harder and harder. Technologies such as Laptops, PDA's, GPRS, Bluetooth, WiFi, Broadband Home Internet Access, VPN and Extranets are all causes of the breakdown of the business network edge. Roaming workers are taking their computers home at the end of the day, it is at this point that around 80-90% (reference this) of company's loose track of what is being done on company equipment. The next working day the systems are brought back into the office plugged back into the network or connect to it by WiFi, while the systems were out of the office they could have picked up any number of viruses or trojans. As soon as they are plugged into the internal network under traditional security setups they have completely side stepped any protection placed on the network to stop the problem getting into the network they are now free to cause havoc without bounds. A very good example of this problem is the code red worm attack on the US Pentagon which was free to spread through the Pentagon's internal network causing havoc for the systems administrators.

There are several companies who make solutions for this sort of problem under a commercial license but as yet there are no Open Source solutions to this problem which can cost huge amounts of money to resolve. Commercial systems such as the ones created by SourceFire Inc are extremely expensive to run and provide little in the way of customability to the purchaser, which is why it was decided to write JxNAP, a distributed platform for building such a system for the Open Source community.

The platform is able to support a whole host of disparate network monitoring applications in one distributed system allowing them all to communicate amongst each other for event correlation and system self maintenance. The following are a few examples of applications that could be hosted on JxNAP:

- Network Asset Tracking
- Network Intruder Detection/Prevention Systems
- Network Anomaly Detection
- Roaming Client Tracking
- IDS Event Threat Level Estimation

2.2 The Solution

JxNAP is a distributed modular network monitoring system designed in such a way that it can be used for almost any purpose, from intruder detection to monitoring the usage of a website. The system is made up of 3 components, the node, the hub and the GUI.

The node is responsible for capturing the traffic on the network and doing the primary analysis of it and then passing the relative information on to the hub. The hub is responsible for collecting the data from

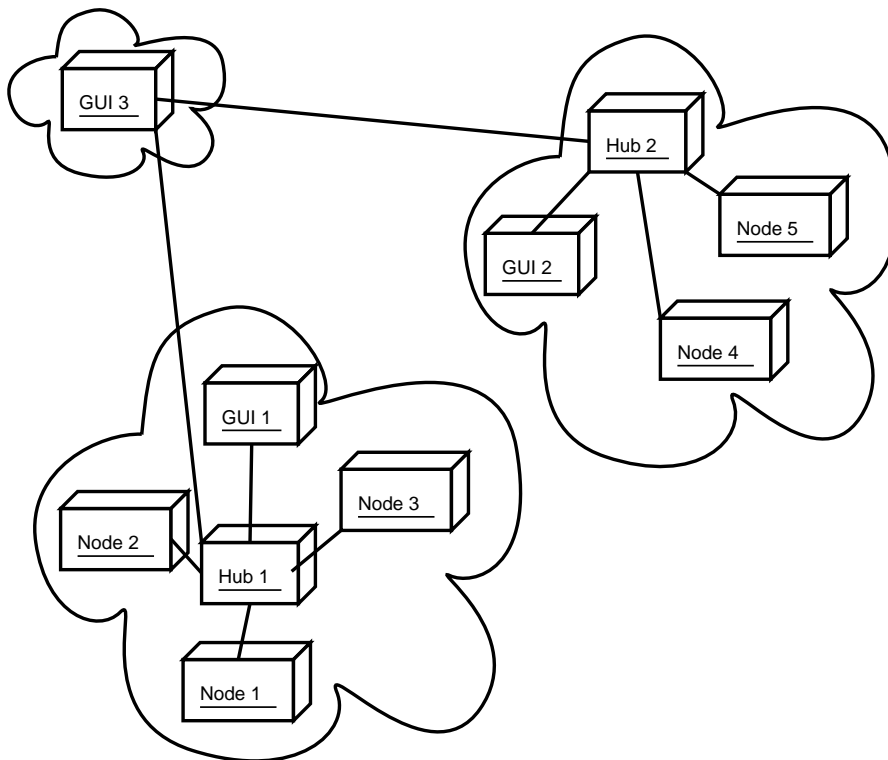


Figure 2: System Over View

several nodes and logging the events in a persistent manner, for later access by the GUI. The hub can also optionally perform further analysis in the event from several nodes and use smaller events from several nodes to for example create a larger more critical event for the entire network being monitored. As well as this the hub is able to maintain logical views of networks by grouping sets of nodes together to form a single network.

The GUI is responsible for displaying the results of the captured data to the network administrators, and generating reports and graphs for them to work from. The GUI will be able to connect to multiple hubs and form an even more powerful view of a single network. The hubs will also be able to support connections from multiple GUI's at any one point in time so that more than one administrator can view and access the information generated.

Figure 2 shows an example of how the parts of the system could be interconnected in an example production environment.

2.3 Project Management

2.3.1 Technologies Used

One of the most common flaws of the open source community is duplicating complete software packages because the only existing packages do not have the feature set that you require. During the course of developing JxNAP it has been decided that instead of writing several components from scratch it would be faster and more efficient to make minor modifications to existing packages rather than implementing the same thing again.

Java

Java was chosen as the main programming language to implement JxNAP in as it opens the project up to a larger target audience, which as not all network administrators are willing to use linux or Unix variants is very important. It also provides a lot of the main API features that other languages don't provide without using large numbers of external library's.

Libpcap and JPCap

Libpcap is the defacto open source packet capture API, unfortunately written in C the JPCap JNI interface allows use of the library in Java. JPCap also supports using the windows port of Libpcap, winpcap, which means that the system can still be used on most unicies and windows.

Libnids

Libnids was incorporated at a very late stage in the development cycle due to a failure to write a functioning TCP stream reassembly engine. Libnids has not been tested on as many operating systems as Libpcap but is based on it, this should mean that I will run on the same platforms that Libpcap and winpcap run on.

OSMQ

OSMQ was chosen as the original base for all inter component communications, however its planned role was reduced during the GUI development phase as it lead to some very difficult asynchronous communication issues. It still provides most of the core functions for the node, hub communications, including dynamic message broker discovery, redundant brokers, automatic transmtion control and object serialisation. After working with it for a prolonged period of time, it has become harder to make effective use of the package due to a lack of good documentation. Any subsequent releases of JxNAP should consider replacing OSMQ with a message broker that is better documented.

JMX

Java Management Extensions (JMX) were chosen as the replacement to OSMQ, for GUI hub communications. JMX provides a RPC³ based interface to management functions of the application being managed. RPC solved a larger number of the asynchronous communication management issues present in the hub. JMX also allows for multiple interfaces to the management extensions, allowing future development to add features such as a web based management console.

MC4j

Management Console 4 Java (MC4j) was chosen for the base of the GUI as it provides a very well documented and stable platform to build management applications that work with JMX. MC4j allows creation of multiple connections to systems that are manageable via JMX. If MC4j was used to manage other applications within the business adding JxNAP's features to it is a very simple job of just adding a new connection.

Log4j

The Apache Foundation's Log4j API could almost be described as the industrial standard logging API for enterprise Java applications. It is very flexible and its output configuration can be changed via a simple configuration file so that system administrators can tune the logging output to their exact needs. It is for these reasons that it was chosen to provide the logging interface for JxNAP.

2.4 Testing

During development regular manual testing took place, as well as a select few automated tests. During the testing stage the 2 nodes were run on the switched network. There was a hub running on a third computer on the network and several GUI connections coming from both the local network and from a remote location via a VPN connection. Alongside this a copy of tcpdump was running on the same hosts as the 2 nodes dumping all of the captured traffic to disc for analysis and checking against the log records created by the system.

2.4.1 Final Timetable

Some items have been swapped around to accommodate the new timeline and some items have been duplicated to dates in the future due to the fact that they were not completed on time and had to be rescheduled for completion at a later date.

³Remote Procedure Call

Week	Task
3	Project Specification Due In Basic System Design Complete
4	Work on basic packet capture interface
5	Basic Packet Capture Interface Complete
6	Implementation of hub core complete
7	Implementation of module loading and module interfaces
8	Hub to node communication protocol designed GUI to Hub communication protocol designed
9	Writing and handing in Progress Report
10	Implementation of hub to node communications Implementation of hub core complete
Holiday 1	Basic GUI Implemented GUI to Hub communication protocol designed
Holiday 2	GUI Module loading implemented
Holiday 3	GUI to hub communications implemented
Holiday 4	Testing of basic system
11	Implementation of first real module
12	Implementation of first real module complete
13	Full testing of system with one module installed
14	Implementation of second module
15	Implementation of second module complete
16	Full testing of system with on demand module loading
17	Provisionally implement a third module
18	Provisionally complete implementation of third module
19	Project Presentation
20	Project Presentation
Holiday 1	Report writing and bug fixing
Holiday 2	Report writing and bug fixing
Holiday 3	Report writing and bug fixing
Holiday 4	Report writing and bug fixing
21	Final report ready for proof reading
22	Final Report deadline

3 The Node

3.1 Analysis Engine

Possibly the most important component of JxNAP the nodes analysis engine is responsible for capturing all of the network traffic, and distributing it to the analysis modules. In order to do this successfully it must be capable of the following:

- Reading Ethernet Frames off the Network
- Determining the payload protocol of Ethernet Frames
- Decoding IP Packets
- Defragmenting IP Packets
- Reassembling TCP Streams
- Determining the operating system of remote hosts

In order to achieve bring this quite significant set of features to the analysis engine the following library's where utilised.

- Libpcap
- Libnids
- Jpcap
- p0f

3.1.1 Capture Engine

The capture engine is a mix of c/c++ and Java code interfaced together with JNI. Figure 3 shows the work flow of the Capture Engine, the capture engine runs in a dedicated thread.

The input from the network card is handled by a combination of Libpcap and Libnids, Libpcap is responsible for capturing individual ethernet frames while Libnids is responsible for capturing and assembling TCP streams. Libnids was included at the very last minute as a simple solution to a very broken TCP stream reassembly engine that was written in Java from scratch. This has led to a somewhat large increase in the compute power required to analyse the network traffic as each frame is effectively being processed twice. Figure 3 shows the Stream assemble as part of the capture engine after Libpcap, which is the original planned stream assembler design. The new stream assembler will be explained in more detail in a later section.

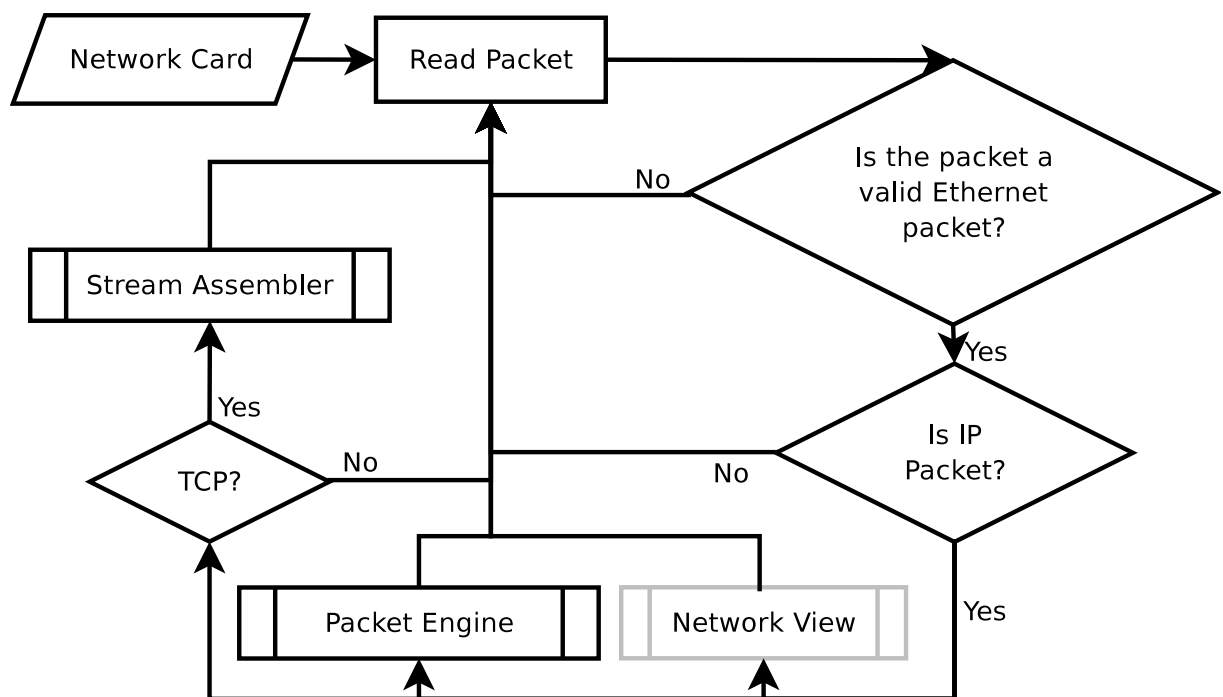


Figure 3: Frame Capture Engine

When a packet is captured by Libpcap it is passed into the JVM via JNI, once inside the JVM all of the analysis work is carried out. The first thing that happens is that JPCap checks and assigns an ethernet frame object to the frame. After this the frame is checked for valid payloads listed in the Ethernet Types[2] document. In the current prototype only ethernet frames with an IPv4 payload are passed onto the packet processing engine for further analysis. At this stage in the original design the IP payload would have been checked for a TCP payload and if so passed on to the TCP stream reassembler, due to the use of Libnids this is no longer required to be done inside the JVM.

3.1.2 Packet Processing Engine

The Packet Processing engine is responsible for doing the basic first level of analysis of all IP packets captured. The packet processing engine is built around a cached thread pool, every time a packet is received the next stage of the packets processing and analysis is delegated to a thread from the thread pool. The use of the thread pool removes some of the overhead of creating new threads for every single packet which on a heavily loaded gigabit network could bring the system to a stand still. Figure 4 shows the work flow of an analysis thread in the packet engine.

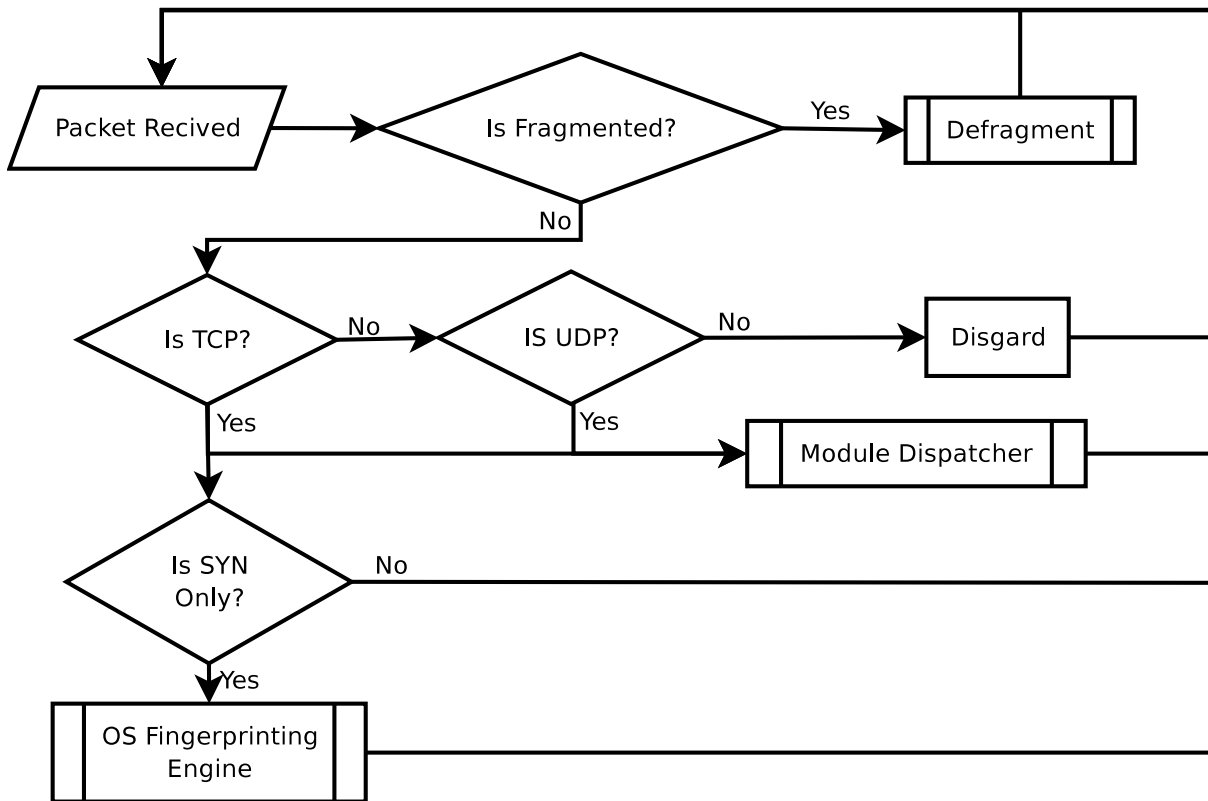


Figure 4: Packet Processing Engine

When a IP packet is received the first thing that happens is that it is check to determin whether or not it is a fragment of a packet. If the packet is fragmented it is sent to the Defragmentation Engine to be defragmented and reassembled for reprocessing. When a unfragmented packet is received the IP payload is checked for know payload protocols (mainly TCP and UDP) and then passed to the module dispatcher. If a TCP packet is received it is also checked to see if it is the first packet in the stream, if it is the packet is also passed to the OS Fingerprinting engine for analysis.

3.1.3 Network View

To provide the traffic analysis modules with more information about the hosts who's traffic they are monitoring, each node will build a topological view of the network they are monitoring from the traffic that is being analysed. Figure 5 shows a simple example network that the node could be running on. Some examples of the data records that could be collected about this network are listed in Table 1.

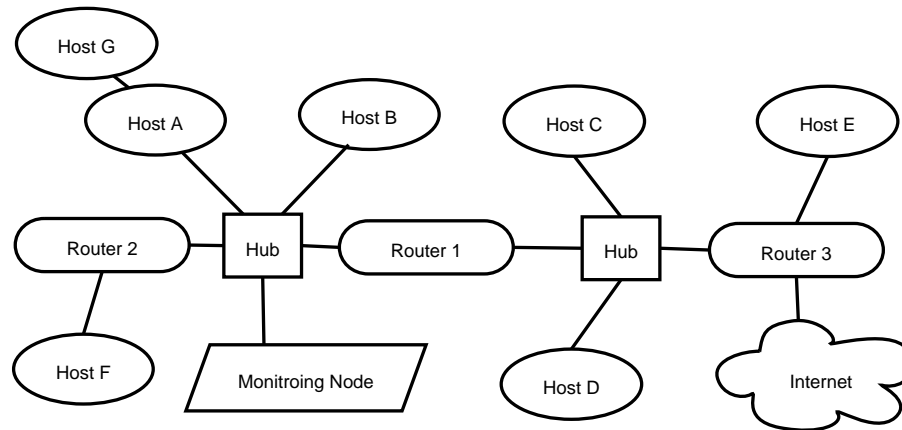


Figure 5: Example Network

Destination IP	Destination Hardware Address	ARP Reply
Host B	Host B	True
Host A	Host A	True
Host C	Router 1	False
Router 1	Router 1	False
Router 1	Router 1	True
Host D	Router 1	False

Table 1: Network Information Records Created From Received Packets

The only real way to build a true topological view of the network being monitored is not only to monitor the IP packets that are being transmitted and received, but to include information collected from protocols such as ARP and RARP. Without this information it is not possible to determine the real IP address belonging to any one IP address. An example of the view of a network constructed without ARP/RARP is shown in Figure 6.

Figure 7 shows a network who's topological view has been constructed from both IP and ARP/RARP data. It is clear that with the extra data it is a lot easier to construct a more realistic topological view of the network being analysed.

Unfortunately the process of maintaining a map such as the one required to represent the structure of a network is is very computationally expensive as for every new piece of information that is received the map could need a untold number mutations to make it correct, for extremely large networks just a metropolitan area network backbones this could mean that the map requires several hundred thousand mutations a second to keep it up to date. Adding this computational workload to a system that is already

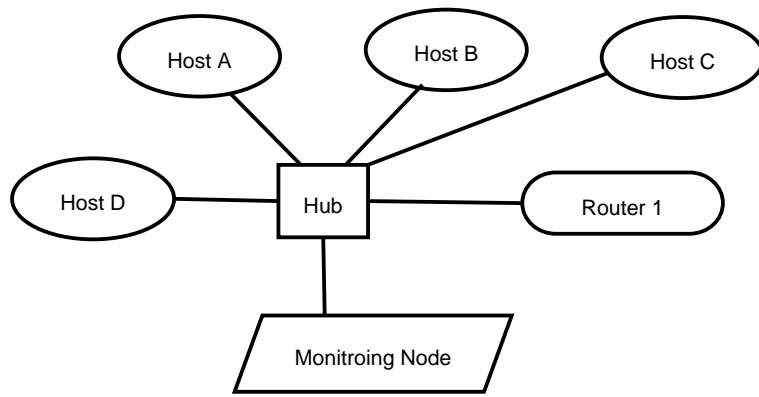


Figure 6: Simple Network

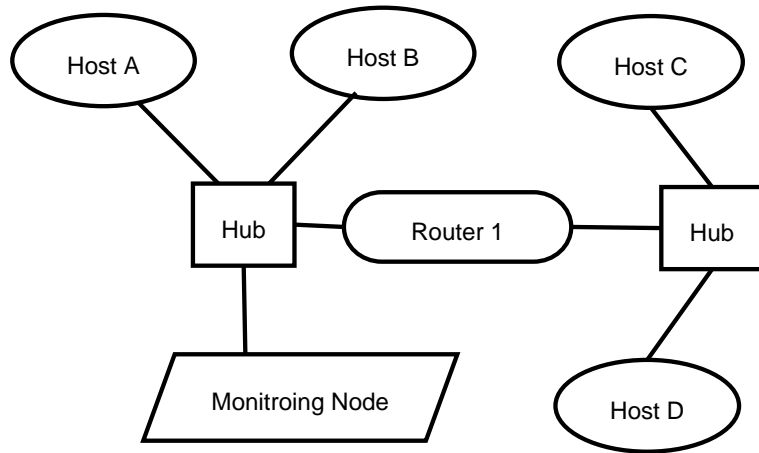


Figure 7: Realistic Network View

stretched to its limits trying to keep up with the traffic on the network simply is not feasible which is why this feature was left out of the node prototype.

3.1.4 Stream Assembler

The job of the stream assembler is to reconstruct the data streams of a TCP session and present them to the analysis modules as two streams (upstream and downstream). Initially planned as a completely Java component after two complete rewrites of the code from scratch it was decided that an external test stream assembler should be used to save time. In the prototype node Libnids is used perform this function, by means of emulating a Linux[14] 2.0 series kernel. Using JNI a small method was written to pass each TCP packet in sequence into the JVM and the stream module dispatcher.

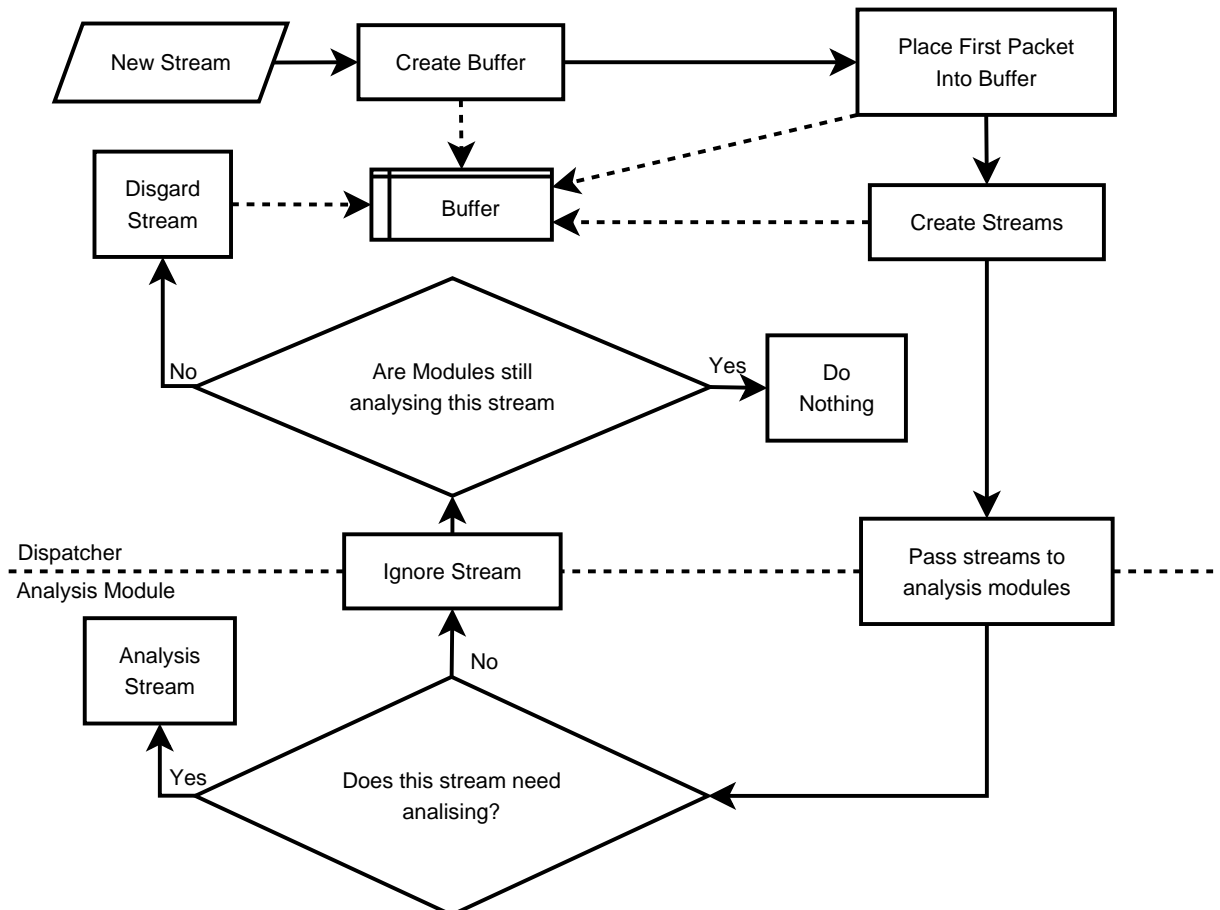


Figure 8: Current Stream Assembler

Figure 8 show the work flow of the stream assembler and dispatcher when a new stream is received. For each subsequent data segment received it is placed into the streams buffer which is available to all of the analysis modules though the IgnorableStream interface. Figure 9 shows the original design for the stream assembler which is no longer included in the prototype.

One thing that has been discovered is that it is very important to reassemble TCP streams in the same manner that the target operating system assembles them. Using rouge TCP packets is a common way to avoid detection on common IDS and IPS systems. These systems may assemble the stream in a Linux fashion where as the system that is being targetted is a Windows system, using this type of evasion it is possible for an attack to pass the IDS/IPS without ever being noticed. JxNAP currently does not assemble streams in a target based fashion due to the lack of support from the OS fingerprinting engine,

as it is not able fingerprint packets other than TCP SYN packets.

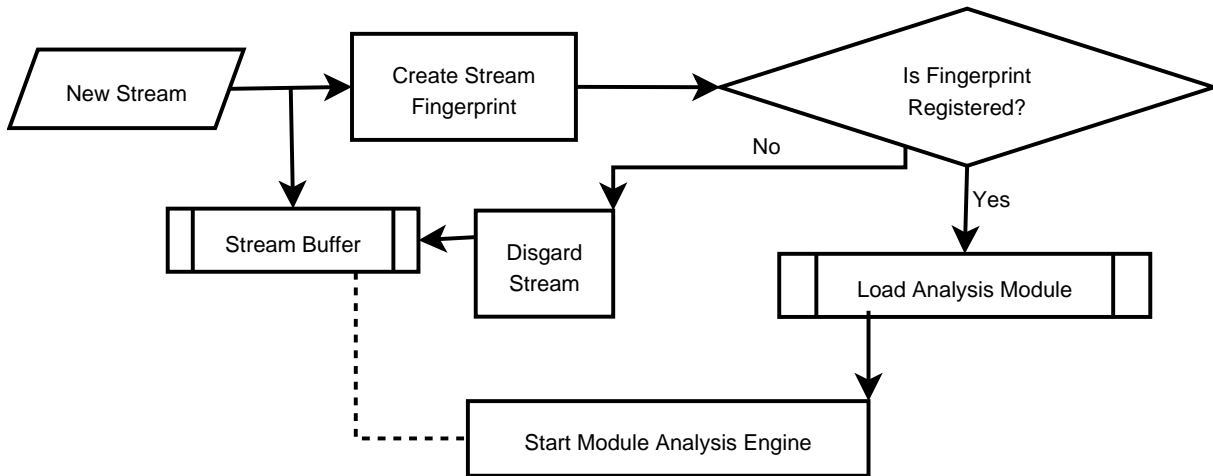


Figure 9: Original Stream Assembler

3.1.5 Defragmentation Engine

Fragmented IP packets normally originate from networks whose maximum frame size is smaller than the IP packets payload, usually because the payload protocol does not support fragmentation (such as UDP). Occasionally IP fragmentation is used as a method of IDS/IPS evasion. It is the job of the defragmentation engine to reassemble IP fragments into complete packets and then push them back into the packet engine for analysis.

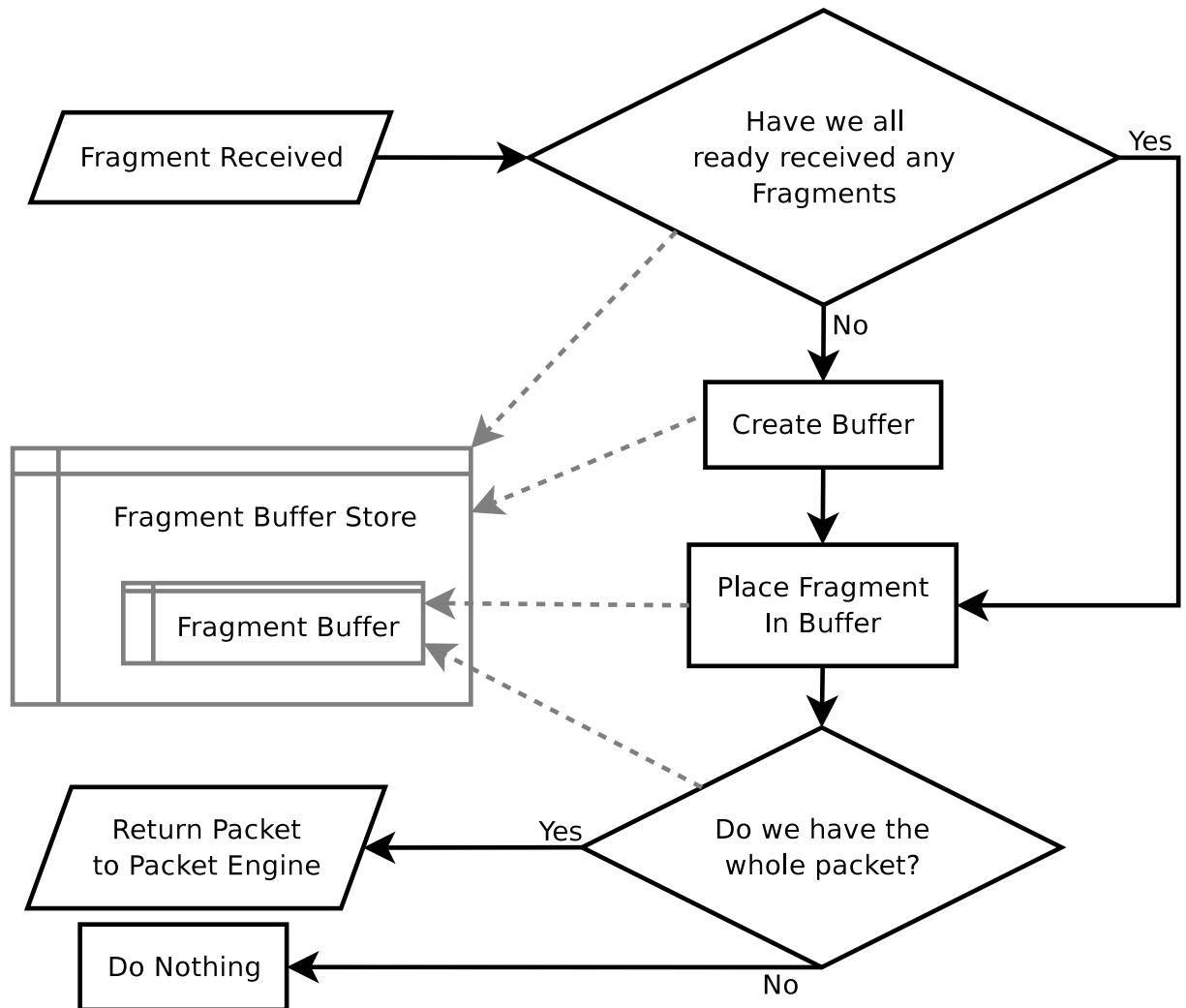


Figure 10: Packet Defragmentation Engine

Figure 10 shows the workflow of the defragmentation engine, which is designed to operate inside the Packet Engines worker threads. To avoid excess memory usage from partial packets being retained in memory each buffer is assigned a timer initially set to count down from 10 minutes, after which time if the packet has not been completely reassembled and returned to the packet engine, all stored fragments are discarded. If a packet is completed within 10 minutes then the timer is cancelled and the completed packet is reassembled and returned to the capture engine, its buffer in the defragmentation engine is discarded.

As with TCP reassembly it is very important defragment packets in the same fashion as target operating system, if this is not done fragmentation can be used to make attacks appear invisible to the system.

Unfortunately the OS fingerprinting engine is not powerful enough to fingerprint UDP packets or any TCP packets other than SYN packets so this feature is not in the prototype node.

3.1.6 Module Dispatcher

The module dispatcher is a very simple component that is responsible for distributing captured packets to the loaded analysis modules. Once the analysis module has received the packet it down to the module to decide whether further processing needs to take place or if the packet is to be ignored. Figure 11 shows the work flow of the dispatcher. It is presumed that the module with start its own worker threads if the analysis is going to take a long time to too help free up the packet engines worker threads.

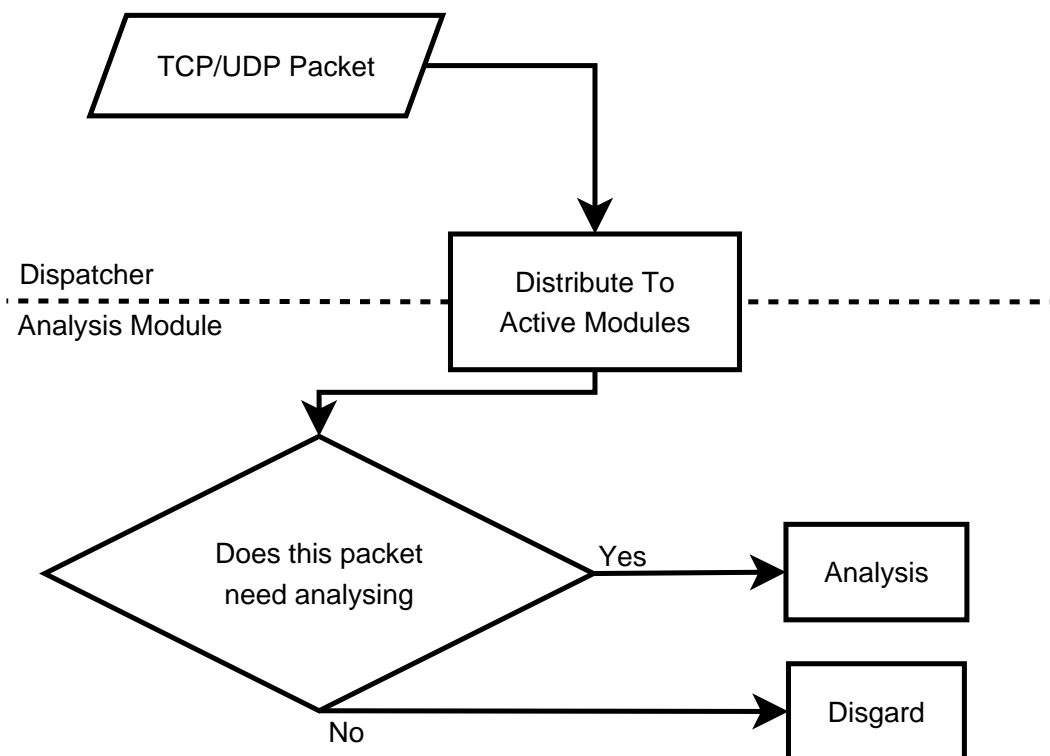


Figure 11: Module Dispatcher

3.1.7 OS Fingerprinting Engine

OS Fingerprinting is a vital part of any advanced network analysis system, without it, it is not possible to perform such functions as target based IP packet defragmentation, or target based TCP stream reassembly. Unfortunately there is not much information about passive OS fingerprinting available online as most systems that accurately fingerprint operating systems do it in an active fashion⁴, in order to reduce the detectability of the system an active scan is not suitable.

The most advanced OS fingerprinting system was written by Martin Roesch⁵ of SourceFire for their RNA sensor product. Unfortunately this is a close source project, but it shows that with work it is possible to fingerprint any common IP payload packet. In order for JxNAP to be a useful system the OS fingerprinting engine will need to be enhanced to this level.

The current system borrows heavily from the p0f[12] project and is mostly a straight port of the C code to Java. Porting p0f[12] to Java meant that the system stays more platform independent even though p0f does have a slave mode allowing packets to be passed to it and it to return the results to the client. Only the most stable analyser (The TCP SYN packet analyser) was ported as a proof of concept for the prototype.

Each operating system fingerprint is stored in a simple text file which is parsed on system startup, the following is a list of information that is used to make up a single operating system's fingerprint.

- IP Information
 - Initial TTL
 - Do not fragment bit (IP)

pagebreak

- TCP Information
 - Window Size
 - Overall SYN Packet Size
 - Options and the Order that they are set in and the values set by them
 - * NOP option
 - * EOL option
 - * Window scaling option
 - * Maximum segment size option
 - * Selective ACK OK
 - * Timestamp
 - * Timestamp with zero value
 - * Unrecognised option number

⁴Nmap is an example of a utility with active OS fingerprinting.

⁵Martin Roesch is also the author of the extremely popular open source IDS Snort.

- Quirks in the TCP/IP stack of the remote host that causes values to be set incorrectly or invalid option configurations.
 - Options past EOL
 - Zero IP ID
 - IP options specified
 - Urg pointer non-zero
 - Unused (x2) field non-zero
 - ACK number non-zero
 - Non-zero second timestamp
 - Unusual flags (PUSH, URG, etc)
 - Data payload
 - Broken options segment

3.1.8 Traffic Analysis Module

The Traffic analysis module records information about the volume of traffic being captured by the system. It records traffic volumes by source and destination TCP and UDP ports and by source and destination IP addresses.

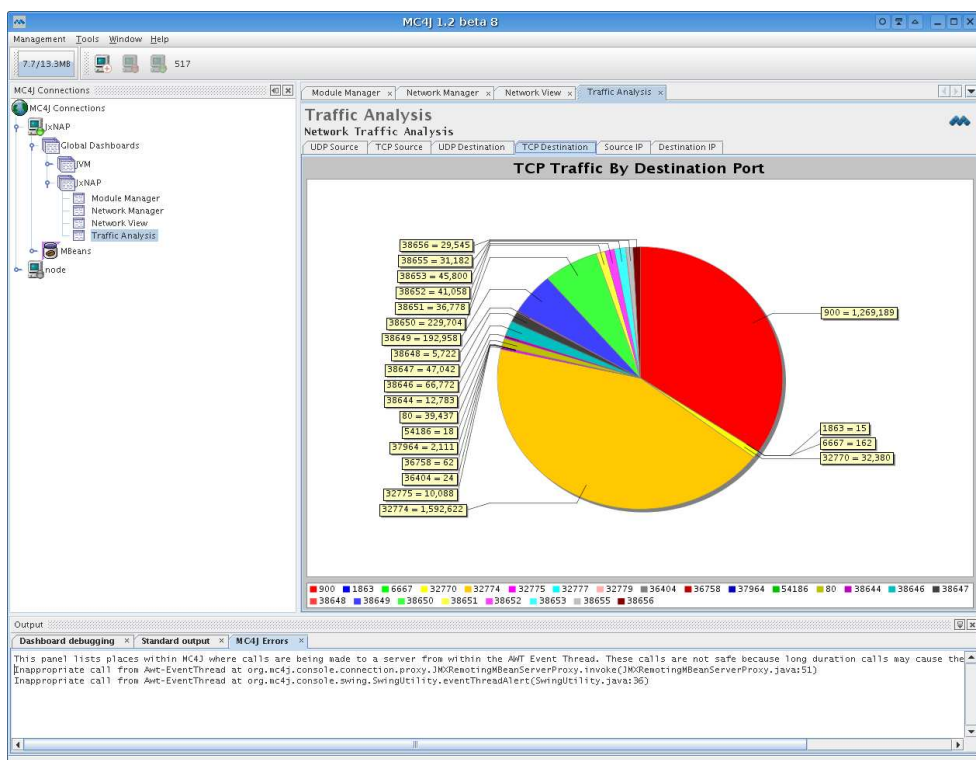


Figure 12: Screenshot showing traffic analysis plugin output

IP traffic volumes are calculated from the Packet Length field of the IP header⁶. TCP/UDP traffic volumes are calculated from the volume of the TCP/UDP⁷ payload data.

3.1.9 MSN Messenger Analysis Module

To show the a security orientated use of JxNAP a module has been written that searches for keywords in MSN Messenger conversations. This could be useful for companies who feel that instant messaging is a useful productivity tool and therefore do not want to block access to it, but do want to be able to keep track of the information that the employees are send over it.

The main part of this module is formed from a heavily cut down version of TjMSNLib[13] which provides the logic for processing commands in sessions between clients and MSN messenger servers. The plugin currently ignores any streams that are not established to a know MSN Switchboard Server port⁸, in order to improve the reliability of the plugin it would be advisable to include a heuristic stream fingerprinting engine to the plugin so that conversations taking place to unusual ports can also be monitored. When the plugin detects a listed keyword in a session it then forwards the entire message to the hub for further processing.

For a demonstration of the MSN Plugin in action please see the demonstration videos included on the CD inside the backcover of this document.

⁶See Figure 21

⁷See Figures 22 and 23

⁸TCP Port 1863

3.2 Hub Communications

The hub communications system is built around the OSMQ system for distributed messaging. Each node uses multicast discovery system to locate the nearest hub to them, upon locating the hub they establish a link to the hub. This connection is then used for all communications between the node, hub, GUI and other nodes.

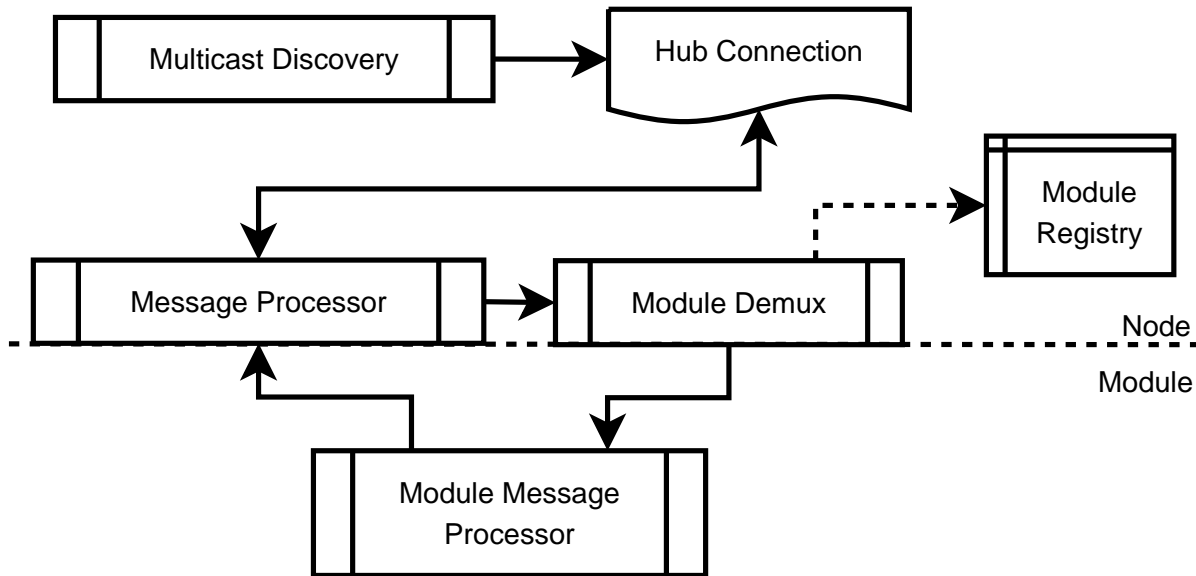


Figure 13: Hub Connection Overview

A single data structure is used for transmission over the communication link, while OSMQ handles serialisation and transmission of each message. The message is a field based data structure that is serialised for transmission over the communications channel. For node to hub communications four message types are defined:

- Status Messages
- Command Messages
- Data Messages
- Acknowledgement Messages

3.2.1 Status Message

The status message only gets transmitted between the node at the hub when the internal state of the node changes. The first field of the status message is the message identification as set by the transmitting node, the second field of the message is the status of the remote node. The node can be in one of three states at any one time Offline, Online or Capturing. The Offline and Capturing state are pretty self explanatory, but Online means that the node is running and accessible over the communications network but it currently no capturing for some reason, normally because it has not yet been configured.

3.2.2 Command Message

The command message is used by either the node or hub to transmit commands to its peer. The first field of the command message is the message identification as set by the transmitting process. The second field of the command message is the command field containing the command to be executed. The third and fourth fields of the message are optional data arguments required for processing the command.

There are currently four commands that are used in node to hub communications:

GET_CONFIG Used by the hub to tell the node to transmit its current configuration to the to the hub.

CONFIG_UPDATE Used by the hub to transfer and activate a new configuration to a particular node.

MODULE_COMMAND Used by modules send to commands between the hub and node resident components of the module.

INSTALL_MODULE Used by the hub to install a new module into an active node.

3.2.3 Data Message

The data message is used by either the node or hub to transmit data to its peer. The first field of the data message is the message identification as set by the transmitting process. The second field of the data message is the data field containing the type of data attached. The third field of the message is the data itself, there is an optional fourth field for a second piece of data if required.

There are currently three data types that are used in node to hub communications:

IDENT Used by the node to identify its self with the hub.

CONFIGURATION Used by the node to send its configuration to the hub after receiving a GET_CONFIG command.

MODULE_DATA Used by modules to send data between the hub and node resident components of the module.

3.2.4 Acknowledgement Message

The acknowledgement message (ACK) is used by both the node and hob to confirm that a message has been received and processed. The first field of ACK message is the identification of the message as set by the sending process. The second field of the ACK message is the sender of the original message, the third field is the ID of the message that is being acknowledged. The ACK message is the only message used that does not have to have an ACK message transmitted by the recipient on receipt of the message.

3.2.5 Node Communication Startup Process

Figure 14 shows the communication sequence as the node starts up and registers itself with the network. The initial Ident message is used to map the random session identification of the node to a system administrator assigned friendly name for the node. As soon as the node becomes fully operational it also sends its status to the hub. When the hub receives the Ident message this triggers the transmission a configuration request, while a node is connected to the network the hub always stores a copy of the active node configuration, to simplify the process of reconfiguring the nodes from a GUI management console and avoiding handling the asynchronous communication between the hub and GUI, while the hub requests the configuration from the node. Upon receipt of the configuration request the node immediately sends a copy of its current active configuration to the hub. Once the configuration is received the hub then sends module installation commands to the node for all of the modules currently installed on the system.

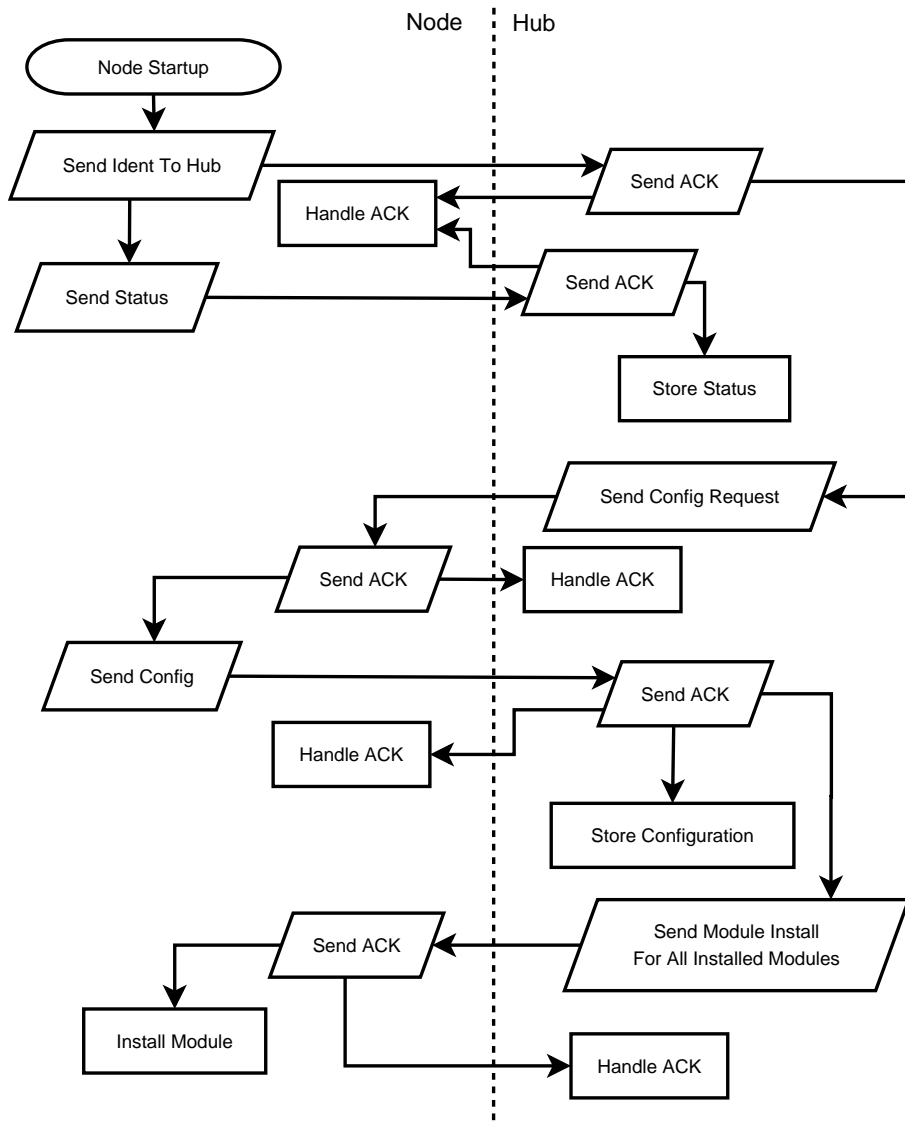


Figure 14: Node Communication Startup Process

3.3 Module Loader and Logging

There are two components that are used in all three parts of JxNAP they are the module loader and the logging facilities. The runtime module loading system used throughout JxNAP is jModuleLoader which loads classes from jar files which are transferred to the node in `INSTALL_MODULE` commands. jModuleLoader was chosen as it provides much of the core functionality for loading modules with very low overheads.

To provide a competent logging system the Apache Foundations Log4j was chose, as it is a very mature and advanced logging system which whose output is completely configurable both at runtime and using a simple external configuration file.

4 The Hub

The hub is the centre piece of the JxNAP system, providing services such as:

- Node and GUI communications management
- System state tracking
- Module event logging
- Active module deployment
- Logical network tracking

Figure 15 shows an overview of the hubs architecture and most major component interaction.

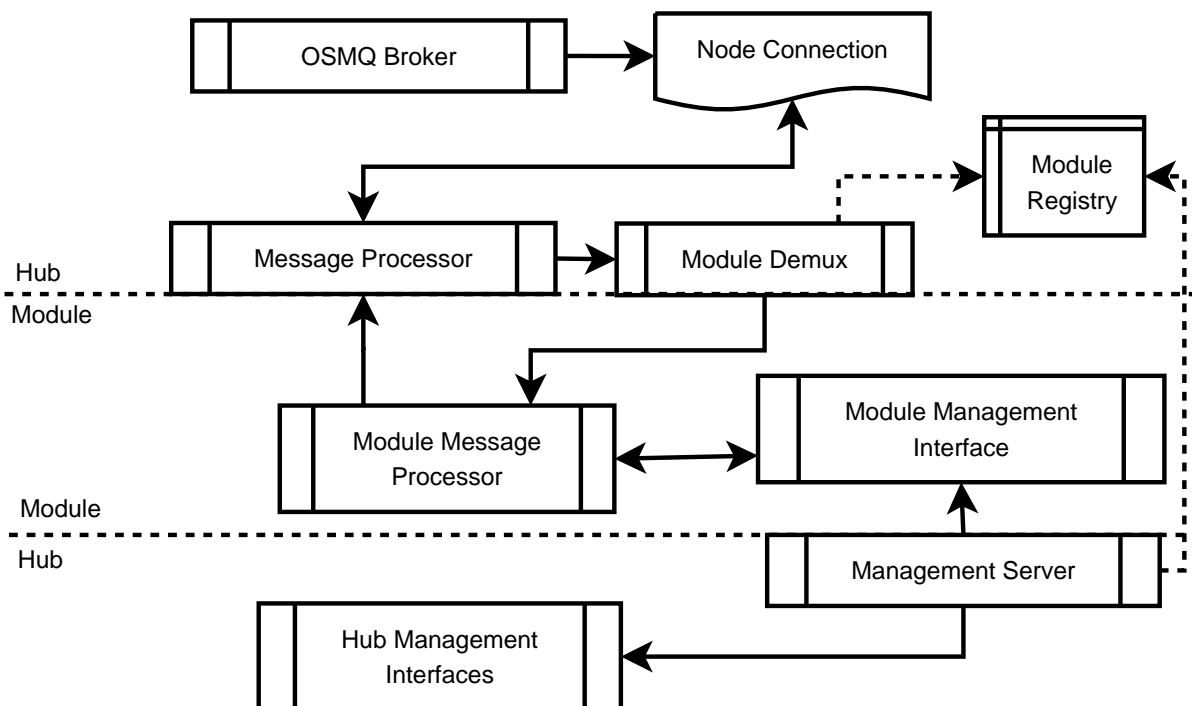


Figure 15: Hub Architecture Overview

4.1 Node Communications

The hub provides the most of the platform for the hub, node communications system. The each hub runs an OSMQ message broker, which in turn provides the dynamic hub discovery services using multicast broadcasting to the nodes. By using multicast rather than standard broadcast the dynamic discovery system can operate across subnet boundaries and over routed networks. Although untested due to a lack of documentation for OSMQ it is possible to have multipul message brokers running on a single network segment. If broker slaving was fully researched and documented it should be possible to use multipul hubs for load balancing a large number of monitoring nodes across several hubs.

The communication protocol between the hub and the node is documented in Section 3.2.

4.2 Persistent Storage

In order for the data recorded by the system to be available after restarting the hub, it requires some form of persistent storage. MySQL was chosen to provide the persistent storage engine for the hub, the MySQL AB provide a standalone server that can be embedded into Java applications call Connector/MXj. Connector/MXj integrates a platform specific MySQL server, which is available for most platforms, with a JMX management interface for controlling the server. By configuring the server to use non standard settings it is possible to have a dedicated MySQL server running for JxNAP that will happily co-exist with a already existing MySQL server. Along side the MySQL server, Connector/j was chosen to provide the JDBC drivers to access the MySQL database from within the application.

In order to improve the efficiency of the storage system in future versions of the hub, use of JDBC connection pooling should be considered, as currently all of the components within the hub share a single connection to the database. This means that only one component can access the storage at any one point in time, which will negatively impact the performance of the system when it becomes heavily loaded.

4.3 GUI Communications

The hub is responsible for providing the main server architecture for the hub GUI communications API. The server components are provided by the Java 1.5/5.0 JMX extensions. Access to the management interfaces is provided by the Java RMI subsystem and the standard JMX RMI server.

The hub has a number of core management interfaces published for basic system management, whose functionality is described in the GUI section, including:

- Node Interface
- Network Interface
- Network View Interface
- Module Interface

5 The GUI

The GUI management console provides almost all of the user interface to the JxNAP system. It provides both system management and monitoring interfaces from the hub which are accessed via JMX. The GUI is built on top of the MC4j platform which itself is build on the Netbeans platform. MC4j provides a very stable method for accessing all of the hubs published management interfaces through Java RMI.

The term used within MC4j for a GUI component that interfaces with a JMX management interface is a Dashboard. There are a number of standard dashboards that are included with the code of JxNAP these provide functions such as logical network management, logical network status and node configuration. Not every published hub management interface has a single Dashboard associated with it, some dashboards access more than one management interface. Each dashboard maintains an up to date view of the management interfaces that it uses by using a timer to periodically update the information from the hub, this allow semi realtime operation of the system, with propagation times at worst up to around 2 minutes and at best a second.

5.1 Network Management Dashboard

The roll of the network management dashboard is to allow control over the hubs internal views of logical networks. The network management dashboard is capable of adding and removing all known nodes from a logical network regardless of the state of the node⁹. A node is not limited to being part of a single logical network, this is useful when for example there is more than one network administrator using the system, each of whom require a different view of the networks security events, but may require their views to acquire data from a common subset of nodes.

5.2 Module Management Dashboard

Interfacing with the module management interface in the hub, the module management dashboard is responsible for installing modules onto the hub for distribution to the rest of the network. The module manager is also able to remove modules from the system but in the current prototype system this requires all of the nodes to restarted. With a very small amount of work it should be possible to add another command to the hub node communication api that causes a module to be unloaded from the node, in order for the node to unload the module the module manager only has to remove all references to the module and then it will be automatically be removed from the JVM during the next garbage collection (GC) cycle. Unfortunately between unloading the module an the GC event it will not be able to reload the same module as all of its classes will still be loaded into the JVM.

The module management dashboard loads the entire module distribution into a special data structure to be transmitted over the network as and argument to an RPC call to the install function on the network.

⁹This function requires that the node must have connected to the hub at least once before the network manager is able to add the node to a logical network.

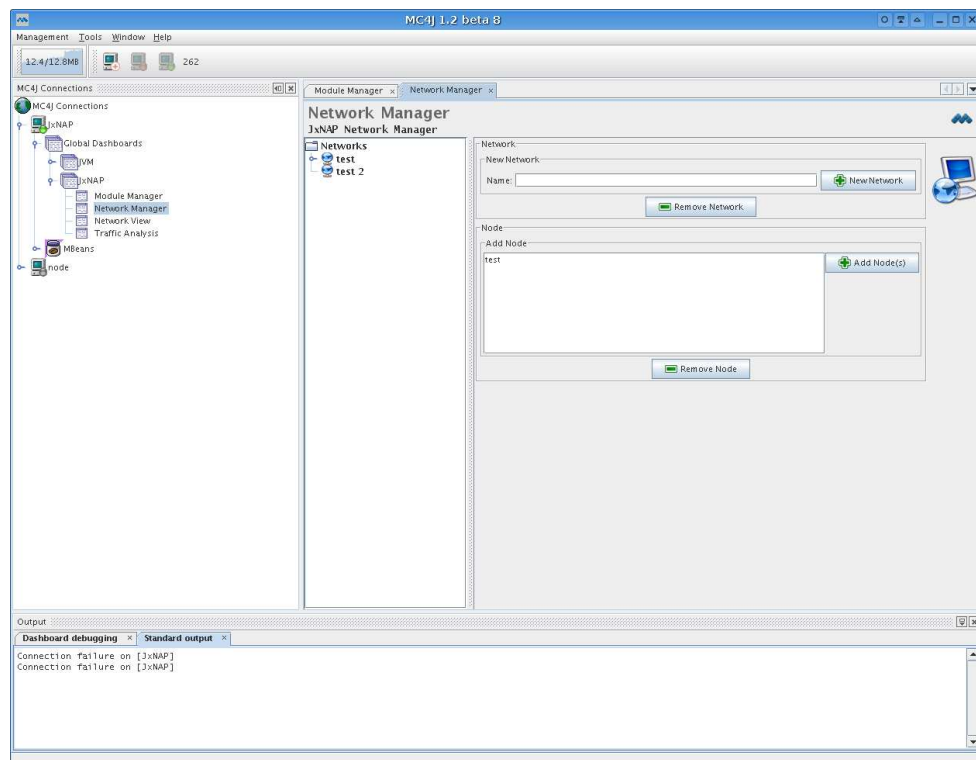


Figure 16: Screenshot showing network management dashboard

5.3 Network View Dashboard

Interfacing with the network and node management interfaces the network view dashboard provides an overview of the state of a logical network on, and allows access to the configuration of the individual nodes. In future versions of JxNAP the network view will also display information from plugins that has been collated and analysed in the context of the logical network.

When configuring the a node the dashboard first retrieves the configuration from the hub using the node management interface, and then loads the configuration into the configuration component. When the user chooses to save the configuration the hub first updates its copy of the nodes configuration and then sends the node the CONFIG_UPDATE command to load the new configuration and restart any required components of the node.

5.4 Traffic Analysis Module Dashboard

The traffic analysis modules dashboard is a little more complex than the other dashboard due to its use of graphs, which are managed and drawn by the jFreeChart API.

5.5 MSN Messenger Module Dashboard

The MSN Messenger dashboard simply lists all messages, that have been flagged up due to them containing defined keywords.

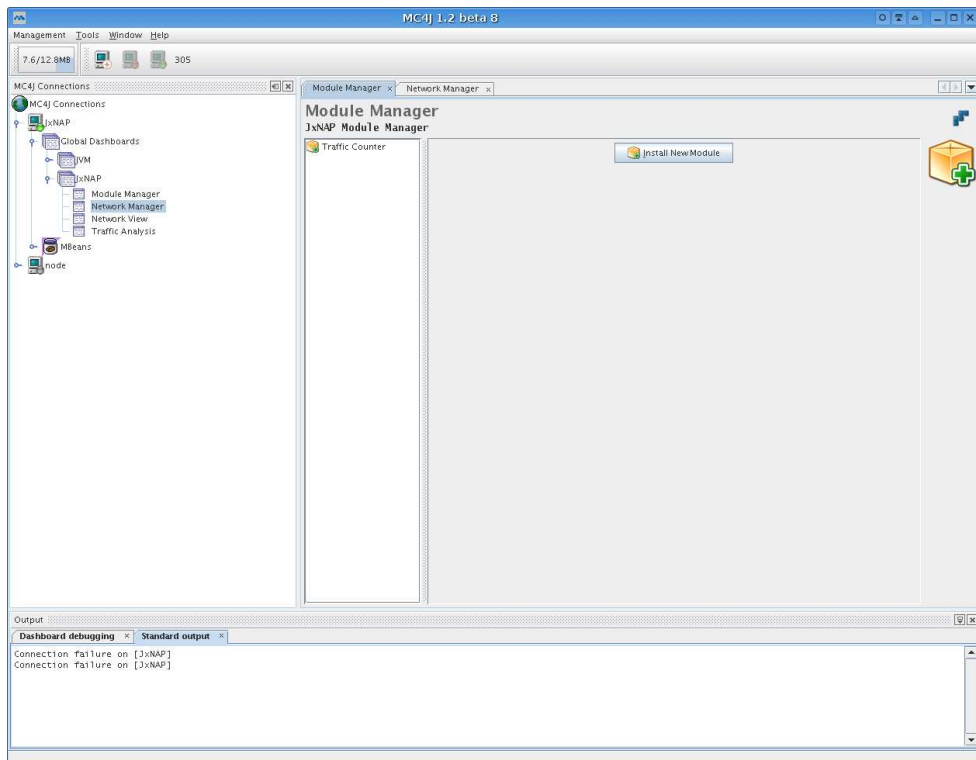


Figure 17: Screenshot showing module management dashboard

6 Conclusion

6.1 Achieved Objectives

Most of the objectives that were set out at the start of this project were reached bar three.

Completed Objectives:

- Provide a stable communication channel between the nodes and hubs
- Provide a stable communication channel between the GUI's and hubs
- Self maintaining packet capture nodes
- Hub based system to simplify system architecture
- Provide information about a host to all analysis modules (e.g. Operating System)
- Provide a stable communication channel between the nodes and hubs
- Modular protocol analysis engine
- Automatic protocol fingerprinting
- Automatic deployment of protocol analysis engines
- Centralised logging of events

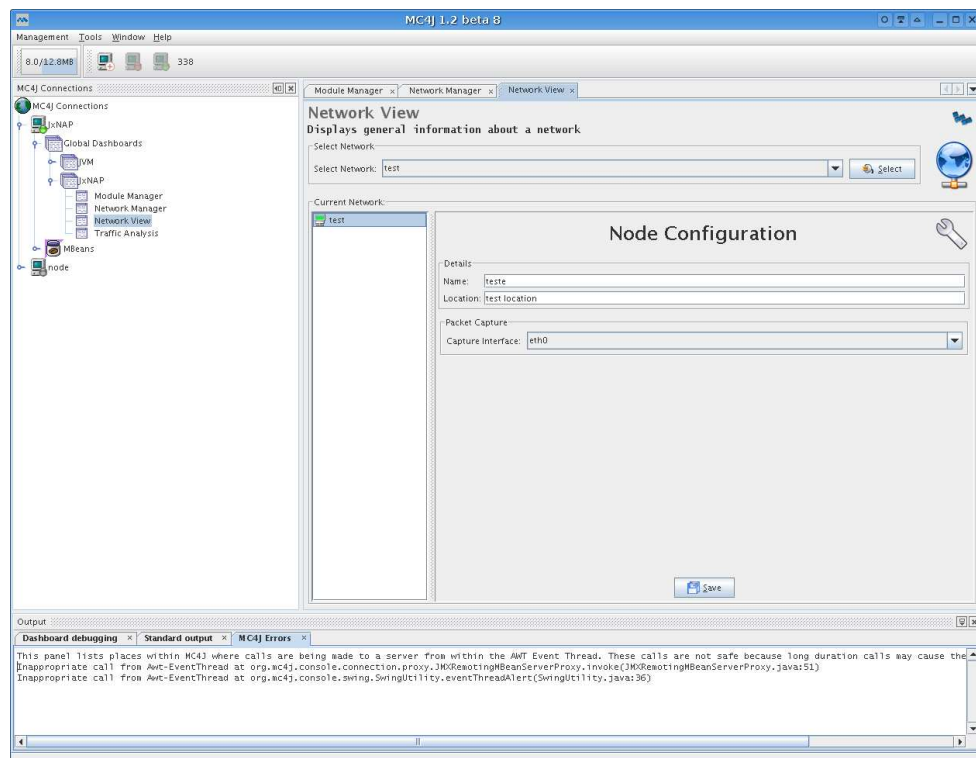


Figure 18: Screenshot showing network view dashboard

- Define networks by grouping nodes from into a single network
- Manage a hub and the installed modules on that hub
- View the status of non event based modules (e.g. a module to passively monitor the usage of a website)
- Provide a stable communication channel between the GUI's and hubs

Failed Objectives:

- Provide a view of the network to all analysis modules
- Provide information about a host to all analysis modules (e.g. Operating System)
- View events from a network or from a specific node

During a refactoring of the topological network view code, it was decided that in order to make the view work on large scale networks a drastically new algorithm for building the view would be required for the node to be able to keep up with the rate at which new hosts could possibly be discovered on the network. It was at this point that analysis modules would no longer have access to the network view. Another reason for not not giving the modules access to the data is that the OS Fingerprinting system is not very accurate. These features would be required to make the platform usable in most network monitoring applications.

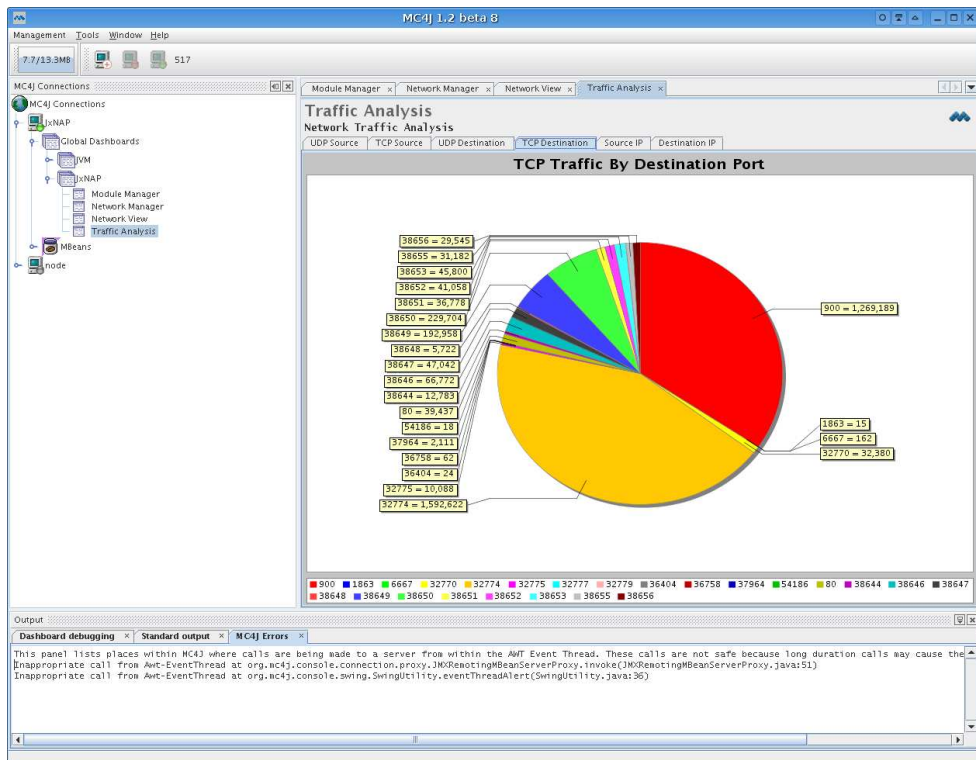


Figure 19: Screenshot showing traffic analysis plugin output

In the prototype each module installed on a network currently publishes a single management interface that is responsible for providing access to the entire module. If the appropriate dashboards and management interfaces where designed modules would be able to present data from the entire system, a particular logical network, or a specific node.

6.2 Difficulties Encountered

There were a few difficulties encountered during the development of JxNAP. The largest single problem was trying to design an asynchronous communication system for the hub to GUI communications to use, however the original design for the protocol was eventually scrapped and replaced with a simpler synchronous protocol. Another major difficulty was writing a fully functional TCP stream reassembly engine, after having written two almost working assemblers and not being able to fully track down the last few bugs it was decided that an external existing assembler should be used to save time.

Although not directly encountered during testing, the lack of OS fingerprinting data and a custom TCP stream reassembler means that there are some quite major security issues with the current prototype. As different operating systems defragment IP packets and reassemble TCP streams in different methods, different operating systems will see different results from IDS evasion techniques. It is for this reason that the IP defragmentor and TCP stream assembler require information about the operating system of the remote hosts in order to assemble the packets/streams being capture in the same way that the destination operating system will do it.

In hind sight choosing OSMQ as the basses for the node hub communication system was a decision.

During later stages of development the lack of documentation for various parts of the system caused some problems and hard to find bugs. If another release of JxNAP was scheduled for a future date, one of the objectives to reach before releasing the new version should be replacing OSMQ with a message brokering system that has better documentation.

A Appendix: Ethernet Frame Header

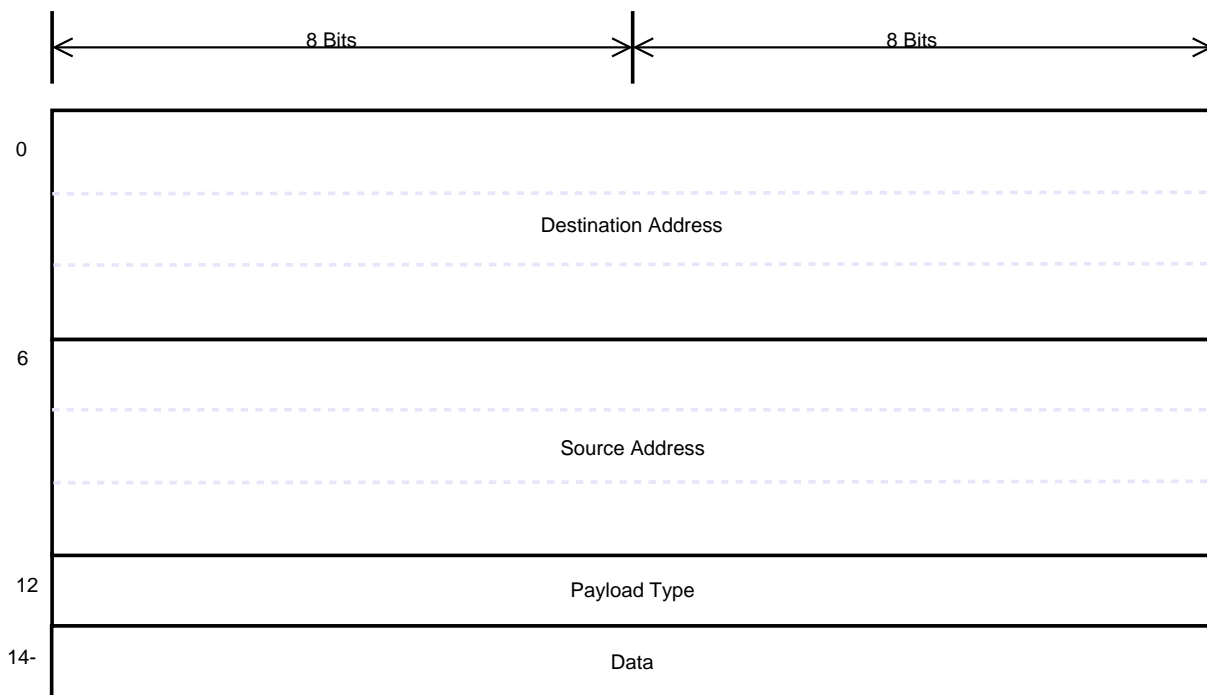


Figure 20: Ethernet Frame Header

Figure 20 shows a simplified Ethernet frame header as used in the JPCap[11] library. Figure 20 is a simplified version of the diagram on page 38 of the IEEE 802.3-2002[1] published standard.

B Appendix: IP Packet Header

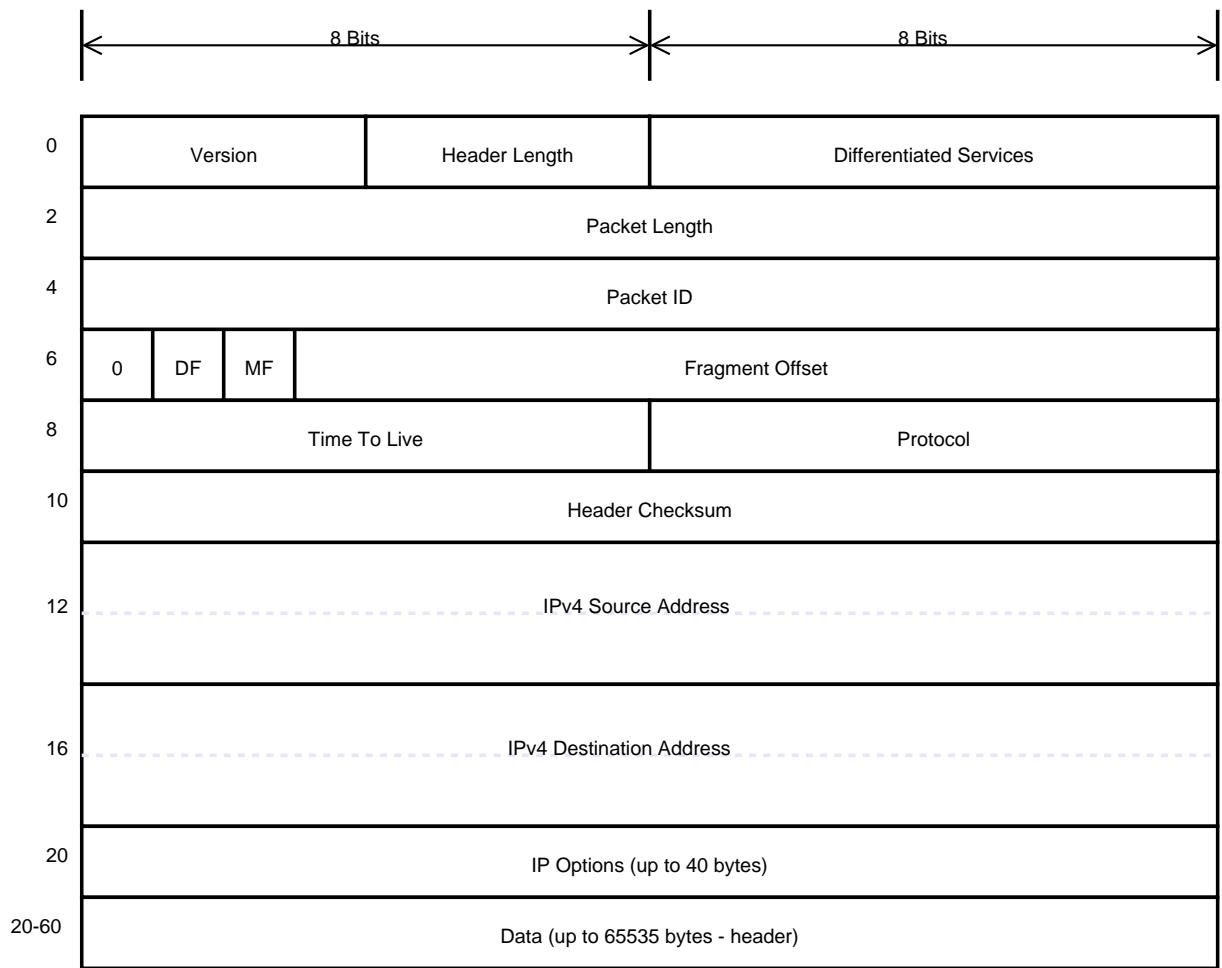


Figure 21: IP Packet Header

C Appendix: TCP Packet Header

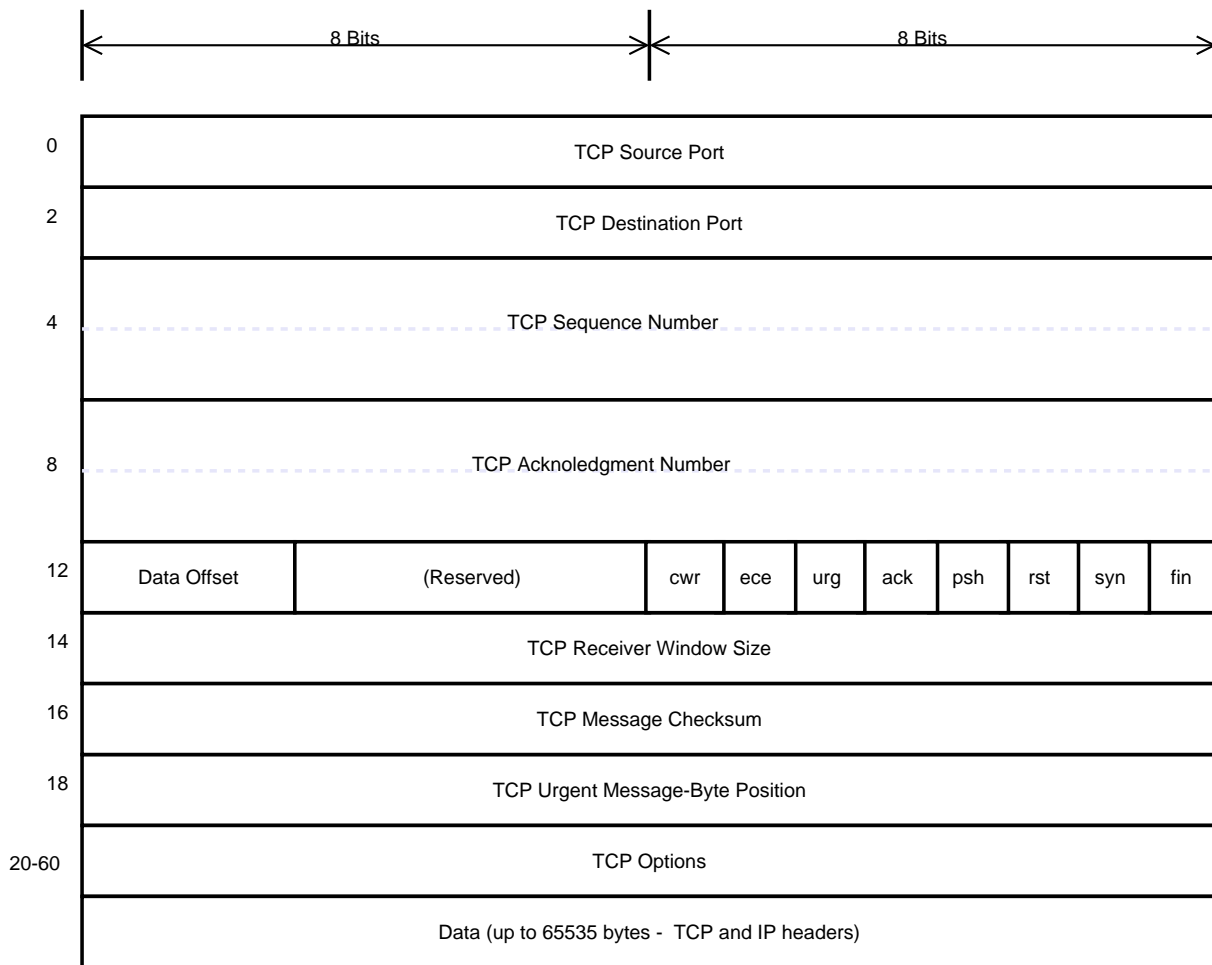


Figure 22: TCP Packet Header

D Appendix: UDP Packet Header

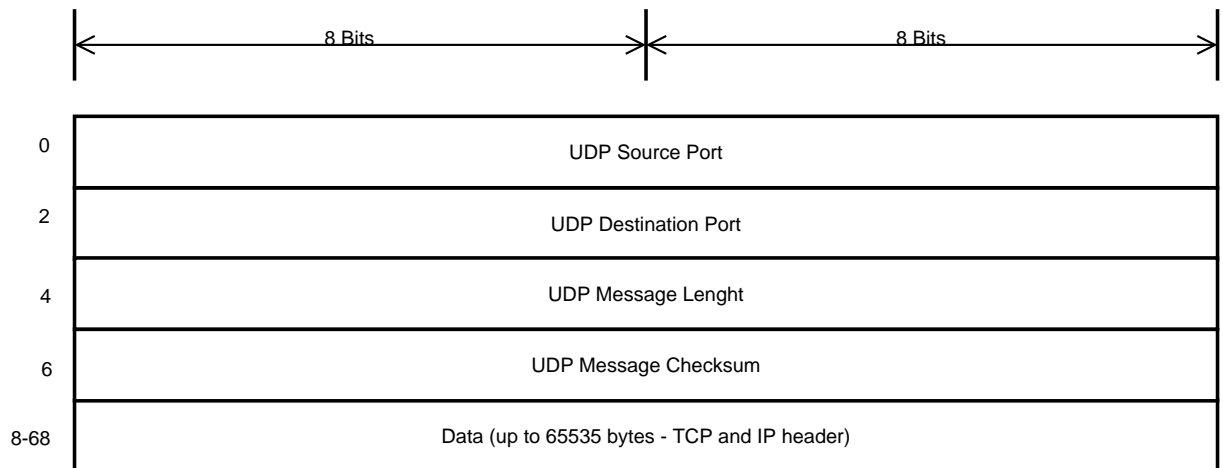


Figure 23: UDP Packet Header

E Appendix: Notes from SourceFire Seminar

E.1 Current Implementation Failings

It is very important that the analysis modules receive the data in a view that would be seen by the applications running on top of the network stack of a host.

The following things would be enhanced/provided to make jxnap suitable for large scale i[dp]s distribution.

- Analysis modules allowed to see raw fragmented data.
- De-fragmentation engine switching to target based de-fragmentation due to different stacks implementing this in different ways, such as segment overlays.
- TCP stream reassembly engine needs to be target based to allow analysis engine to see host view of the packet.
- POF database is not stable and of high enough quality for production systems, ideal solution would be to write a heuristic OS detection engine that could also fingerprint UDP packets as that in sourcefire's RNA sensor can.
- Processing offload to other nodes via a load balancing system.
- Clustering of hubs to allow further scaling of system.
- Gigabit data acquisition is highly processor intensive so the node needs to be able to take advantage of the extra CPU power of the host system.

E.2 Features requested by prospective customers

- SSL connection, tunnelled data analysis, this is not possible it is down to the security system designer to install points in the network where SSL data can be seen in clear text, for example a https proxy server.
- Stream recall, when an event is detected it would be best if the system could recall all of the data from the TCP stream prior to the event. The main problem with this is that for example on a heavily loaded network gigabit this could see the node writing 700-800Mbits/s to disk for recall at a later point in time, or for deletion at another point in time.
- The data acquisition subsystem of the node should be modular so that for example the scanner could run in inline mode or use specialised pieces of network hardware for packet capture.
- Being able to map the network passively is a very important feature.

F Appendix: Bibliography

References

- [1] IEEE 802.3 CSMA/CD Ethernet Standard
- [2] The Ether Types document available at:
*<http://www.cavebear.com/CaveBear/Ethernet/> or
<ftp://ftp.cavebear.com/pub/Ethernet-codes>*
- [3] RFC 791 - Internet Protocol
- [4] RFC 1042 - Standard for the Transmissions of IP Datagrams over IEEE 802 Networks
- [5] RFC 903 - Reverse Address Resolution Protocol
- [6] RFC 768 - User Datagram Protocol - J. Postel 28th August 1980
- [7] RFC 793 - Transmission Control Protocol - September 1981
- [8] RFC 1700 - Assigned Numbers - October 1994
- [9] LIBPCAP Packet capture API (for *nix)
<HTTP://www.tcpdump.org>
- [10] LIBNIDS IDS Orientated Packet capture API (for *nix)
<HTTP://libnids.sourceforge.net>
- [11] jPCAP Java Native Interface to LIBPCAP and WinPCAP
<HTTP://jpcap.sourceforge.net>
- [12] p0f - A passive OS fingerprinting engine
<HTTP://lcamtuf.coredump.cx/p0f.shtml>
- [13] TjMSNLib - A MSN Messenger Library written in java *<HTTP://tjmsn.tomjudge.com/>*
- [14] The Linux Kernel *<HTTP://www.kernel.org>*
- [15] Open Source Message Queue
<HTTP://www.osmq.org/>
- [16] MySQL
<HTTP://www.mysql.com/>
- [17] SubVersion revision control software
<HTTP://subversion.tigris.org>
- [18] Sun Java SDK
<HTTP://java.sun.com>

[19] Netbeans Java IDE

[HTTP://www.netbeans.org](http://www.netbeans.org)

[20] WinPCAP Packet capture API (for windows)

[HTTP://winpcap.polito.it](http://winpcap.polito.it)

[21] \LaTeX and Kile document preparation system

[HTTP://www.latex-project.org/](http://www.latex-project.org/) [HTTP://kile.sourceforge.net](http://kile.sourceforge.net)

[22] jFreeChart a Java chart/graph engine

[HTTP://www.jfree.org](http://www.jfree.org)

[23] Linux Socket Programming by Sean Walton published by SAMS

G Appendix: On The CD

The CD attached inside the rear cover of this document contains the following:

- Documents
 - 802.3-2002.pdf - The IEEE 802.3 specification
 - ethernet-number.txt - Ethernet Types
 - rfc791.pdf - RFC 791 - Internet Protocol
 - rfc903.pdf - RFC 903 - Reverse Address Resolution Protocol
 - rfc1042.pdf - RFC 1042 - Transmission of IP Datagrams over IEEE 802 Networks
 - rfc768.pdf - RFC 768 - User Datagram Protocol
 - rfc793.pdf - RFC 793 - Transmission Control Protocol
 - rfc1700.pdf RFC 1700 - Assigned Numbers
- Research - Various documents and sources
- Source
 - gui-bits - GUI design utility classes
 - jxnap-common - Common component source code
 - jxnap-node - Node Source Code
 - jxnap-hub - Hub Source Code
 - jxnap-gui - Gui Source Code
 - jxnap-traffic-volume-plugin - Traffic Analysis Plugin
 - jxnap-msn-plugin - MSN Messenger Plugin
 - jpcap - Extended Libcap JNI interface with Libnids
 - other - Other library's need to run JxNAP
- Distribution
 - gui-bits.jar - GUI Utility Classes
 - jxnap-common.jar - Common Component Classes
 - jxnap-node.jar - Node Classes
 - jxnap-hub.jar - Hub Classes
 - jxnap-gui.jar - Gui Classes
 - lib - Build of jpcap for linux
- Demos

- node-first.swf - First time node startup
- gui-features.swf - Walk though gui
- msn-demo.swf - Demonstration of the MSN Messenger plugin
- traffic-demo.swf - Demonstration of traffic analysis plugin

G.1 Building from source

Each of the individual components is stored on the CD in a Netbeans 4 project. In order to compile the components, open all of the projects in netbeans and resolve any broken dependencies. Now either Netbeans or ant can be used compile the source code. For convenience I have included a compiled copy of the system on the cd as well.

In order to compile the Gui code a copy of MC4j must be installed, which is included on the CD.

H Appendix: GNU Free Documentation License

Version 1.2, November 2002

Copyright ©2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any

mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **”Invariant Sections”** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **”Cover Texts”** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **”Transparent”** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not **”Transparent”** is called **”Opaque”**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **”Title Page”** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, **”Title Page”** means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section **”Entitled XYZ”** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **”Acknowledgements”**, **”Dedications”**, **”Endorsements”**, or **”History”**.) To **”Preserve the Title”** of such a section when you modify the Document means that it remains a section **”Entitled XYZ”** according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is

included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright ©DATE YOUR NAME Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

I Appendix: The GNU General Public License

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents

or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program’s name and a brief idea of what it does.

Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.

This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program

‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.