

Progress Report: Passive Distributed Network Analysis Using Remote Packet Capture In Java

Thomas C.A. Judge

June 18, 2005

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 else where in the organisation. By monitoring this content in a passive fashion it is possible to monitor systems without touching or reconfiguring them.

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. The project has been named JXNap.

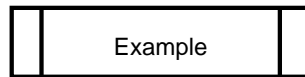


Figure 1: Predefined Process

Contents

1	What Is JXNap	1
2	Current Status	2
2.1	Node Design	2
2.1.1	Network View	3
2.1.2	Operating System Fingerprinting	4
2.2	Hub Design	6
2.3	GUI Design	6
3	Current Status	7
3.1	Completed Objectives	7
3.2	Active Objectives (Currently Being Worked On)	7
3.3	Future Objectives	7
3.4	Revised Timetable	8
3.5	Testing	8
4	Appendix A: Bibliography	9
5	Appendix B: Specification	10

List of Figures

1	Predefined Process	i
2	System Over View	1
3	Node Input Flow	2
4	New Stream Received	3
5	Example Network	3
6	Simple Network	4
7	Realistic Network View	4
8	Packet Headers	5

List of Tables

1	Network Information Records Created From Received Packets	3
---	---------------------------------------------------------------------	---

1 What Is JXNap

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 several hubs and logging the events in a persistent manner, for later access by the GUI. The hub can also optionally perform further analysis in the events 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.

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 multipul hubs and form a 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.

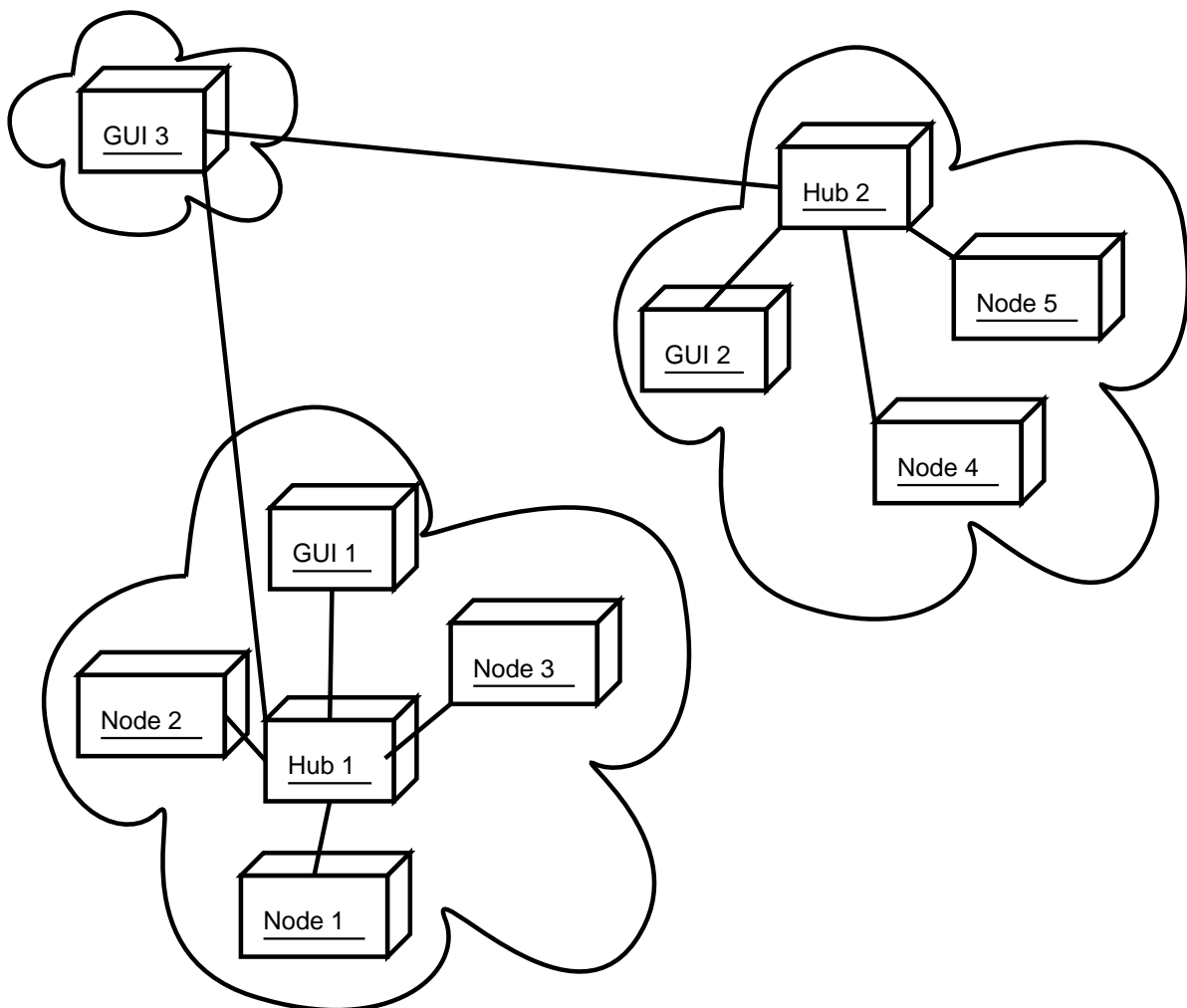


Figure 2: System Over View

2 Current Status

2.1 Node Design

The node is responsible for capturing and doing the majority of the packet analysis work before passing events to the hub. The packet capture side of the node is shown in Figure 3. This shows the route that a packet takes through the logic before ending up in either the Packet Buffer or the Stream Assembly engine. Once received by the Stream Assembly Engine the flow follows the path shown in Figure 4 through the stream assembly engine.

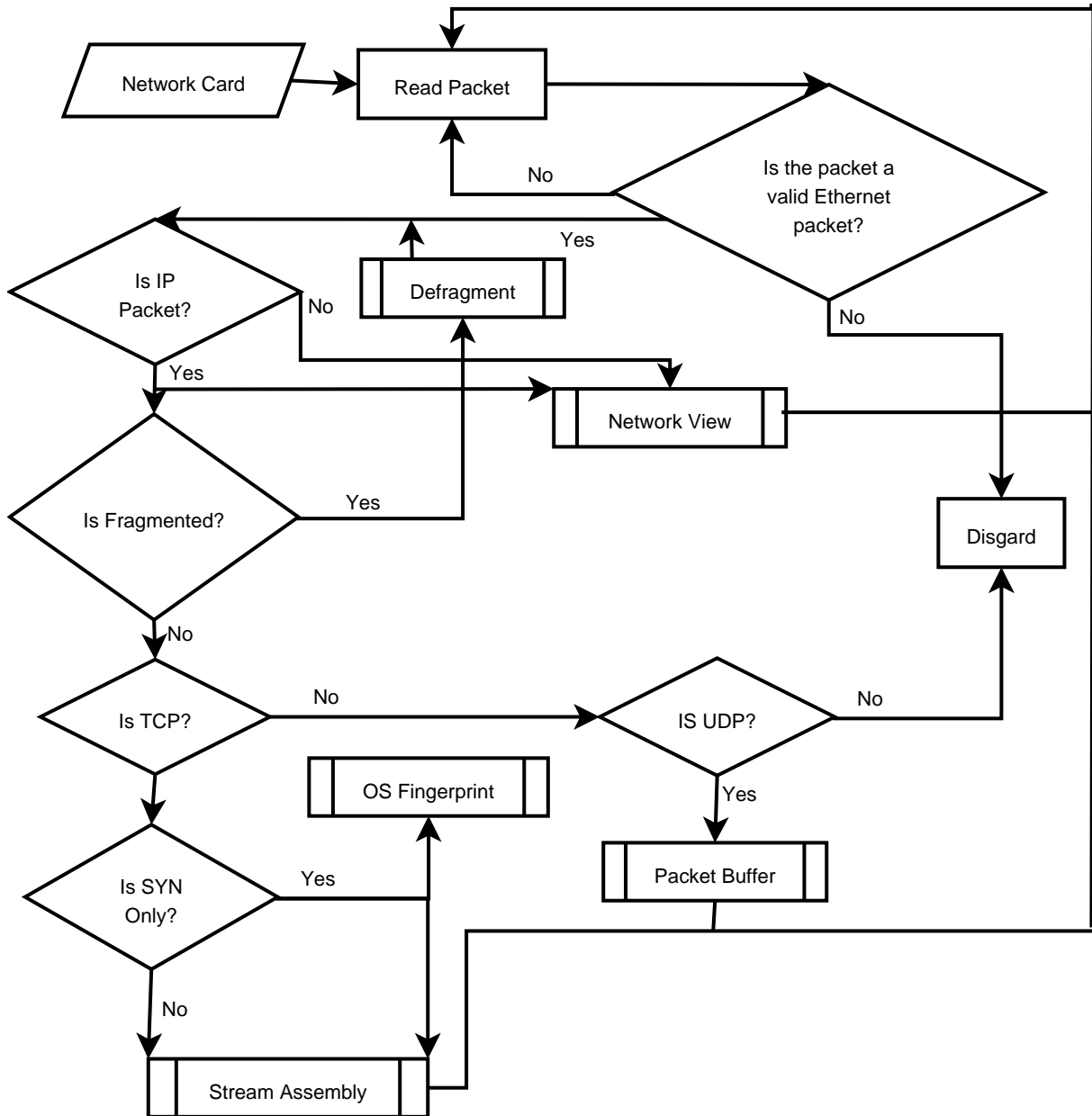


Figure 3: Node Input Flow

The module analysis engine is then responsible for generating events and passing them via the defined interfaces to the hub for logging and further analysis.

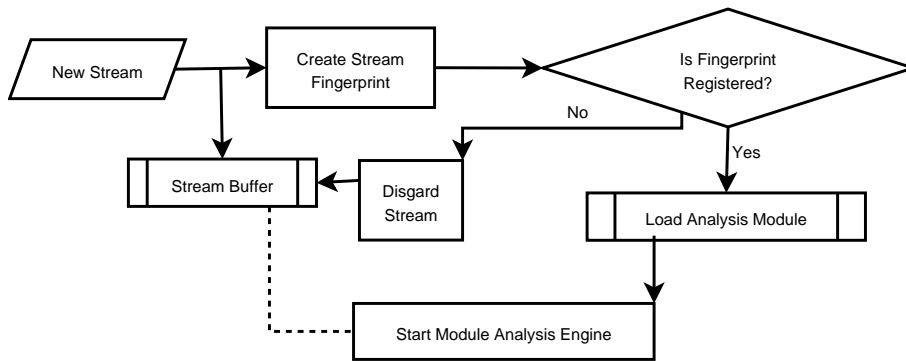


Figure 4: New Stream Received

2.1.1 Network View

To provide the analysis modules with more information about the hosts between which the traffic is passing, the node will build a view of the network from the traffic that it is receiving. 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.

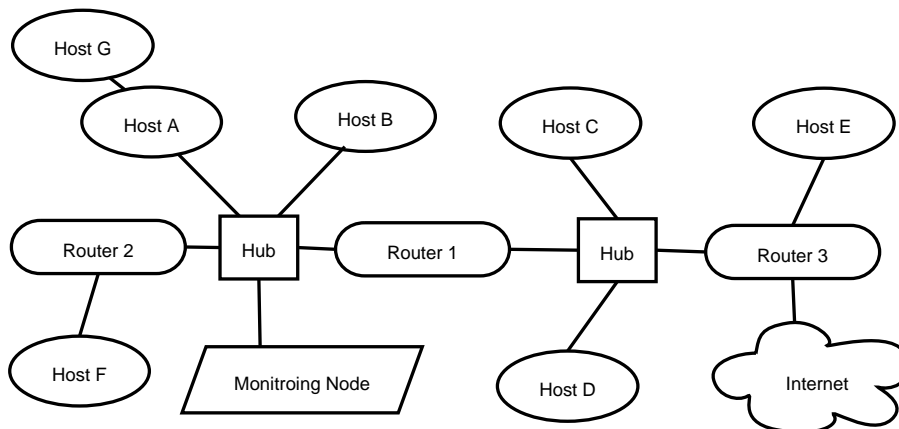


Figure 5: Example Network

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

Without using any of the information gathered from listening to ARP requests and replies on the network the only view that the node would have is one like shown in Figure 6. But by making the network view aware both ARP requests and replies, and of the way in which IP packets are routed between physical network segments it is possible to construct a more realistic view of the networks structure. Using the new process it is possible to construct a view of the network similar to that in Figure 7.

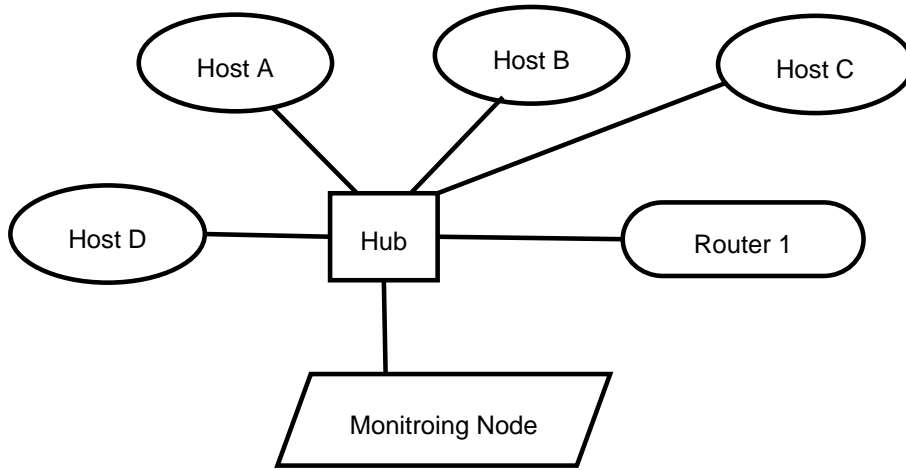


Figure 6: Simple Network

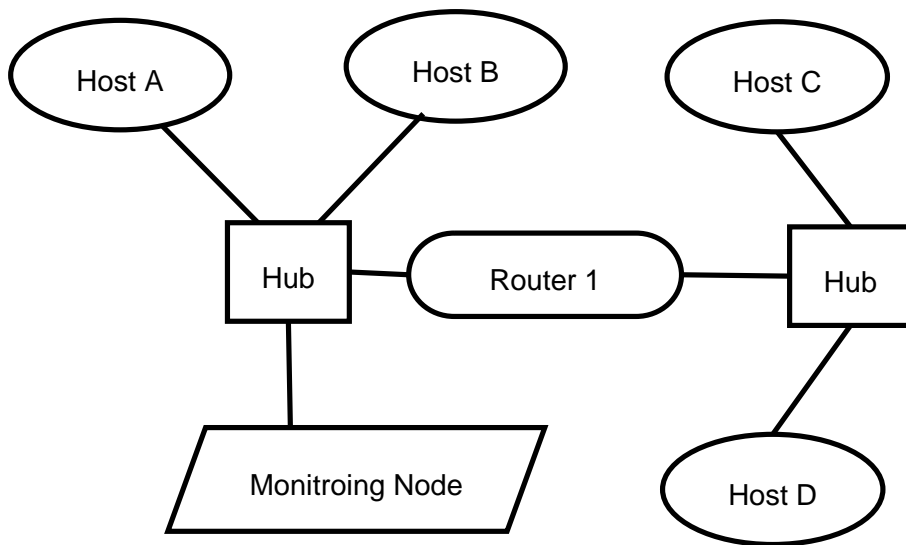


Figure 7: Realistic Network View

2.1.2 Operating System Fingerprinting

The JXNap nodes are now capable of detecting the operating system of a host who's TCP traffic it can see with out creating any traffic between it and the remote host. This is achieved with the help of p0f[3]. I have ported the database and fingerprint from the p0f[3] project to Java along with the matching algorithm.

Operating system fingerprints are made up of the following information which can be found in the TCP and IP packet headers shown in Figure 8:

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

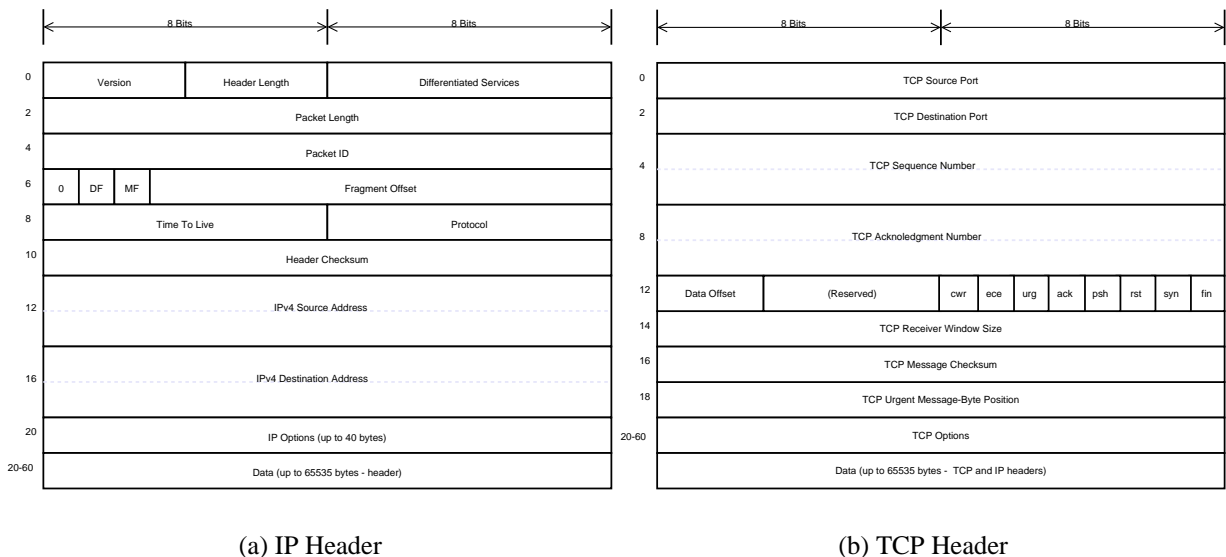


Figure 8: Packet Headers

2.2 Hub Design

The hub will consist of an Open Source Message Queue[2] (OSMQ) broker. This system runs on a free multicast network. The hub will then subscribe to the defined topic of messages that are destined for the hub. Each node will subscribe to the topic for messages destined to the node. This takes much of the underlying communication problems away from the scope of this project.

Each hub will have a logging engine that connects to a MySQL[1] database backend to store event information in. The hub will also run higher level analysis code to tie together information from incoming events from the nodes. The post processed events will also be stored to the logging engine for backup purposes.

The hub will be responsible for distributing active analysis code modules to all the active nodes that are using it as a logging target. As such they will have a cache of protocol fingerprints that are enabled which the node will be able to synchronise with. When the nodes receive a match for the fingerprint they will request the matching analysis module from the hub and install it locally.

The hub will also have a communication system that will enable GUI's to connect to it and request data from the logs. This will be detailed in Section 2.3.

2.3 GUI Design

In the GUI there will be a modular reporting engine for generating reports on the current network activity. This may be based on jFreeReport[11] and jFreeGraph[12]. The GUI will have a facility create new report layouts using jFreeDesigner[13] to create the layouts. The GUI will allow the user to group nodes from many hubs into networks, these networks may not all have nodes on the same hub. From these networks it will be possible to generate network wide reports of all of the activity, as well as hub wide reports and reports on a single node. The GUI will also be able to build a view of a network from the nodes grouped in its internal network views.

3 Current Status

New Objectives (Not Listed In Project Specification)

- Provide a view of the network to all analysis modules
- Provide information about a host to all analysis modules (e.g. Operating System)
- Provide a stable communication channel between the nodes and hubs
- Provide a stable communication channel between the GUI's and hubs

3.1 Completed Objectives

- Self maintaining packet capture nodes
- Hub based system to simplify system architecture
- Provide a view of the network to all analysis modules
- Provide information about a host to all analysis modules (e.g. Operating System)
- Provide a stable communication channel between the nodes and hubs

3.2 Active Objectives (Currently Being Worked On)

- Modular protocol analysis engine
- Automatic protocol fingerprinting
- Automatic deployment of protocol analysis engines

3.3 Future Objectives

- Centralised logging of events
- Define networks by grouping nodes from several hubs into a single network
- View events from a network or from a specific node
- 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

3.4 Revised Timetable

Struck though items are complete. Some items have been swap around to accommodate the new timeline. Some items have been duplicated to dates in the future due to the fact that they were not completed on time and have now been scheduled for completion at a later data.

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

3.5 Testing

During the testing stage the a 2 nodes will be run on the switched network in my house. There will be 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. Along side this a copy of tcpdump will be 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.

4 Appendix A: Bibliography

References

- [1] MySQL
[HTTP://www.mysql.com/](http://www.mysql.com/)
- [2] Open Source Message Queue
[HTTP://www.osmq.org/](http://www.osmq.org/)
- [3] p0f - A passive OS fingerprinting engine
[HTTP://lcamtuf.coredump.cx/p0f.shtml](http://lcamtuf.coredump.cx/p0f.shtml)
- [4] SubVersion revision control software
[HTTP://subversion.tigris.org](http://subversion.tigris.org)
- [5] Blackdown Java SDK
[HTTP://www.blackdown.org](http://www.blackdown.org)
- [6] Netbeans Java IDE
[HTTP://www.netbeans.org](http://www.netbeans.org)
- [7] LIBPCAP Packet capture API (for *nix)
[HTTP://www.tcpdump.org](http://www.tcpdump.org)
- [8] WinPCAP Packet capture API (for windows)
[HTTP://winpcap.polito.it](http://winpcap.polito.it)
- [9] jPCAP Java Native Interface to LIBPCAP and WinPCAP
[HTTP://jpcap.sourceforge.net](http://jpcap.sourceforge.net)
- [10] L^AT_EX and Kile document preparation system
[HTTP://www.latex-project.org/](http://www.latex-project.org/) [HTTP://kile.sourceforge.net](http://kile.sourceforge.net)
- [11] jFreeReport a Java report generating engine
[HTTP://www.jfree.org](http://www.jfree.org)
- [12] jFreeChart a Java chart/graph engine
[HTTP://www.jfree.org](http://www.jfree.org)
- [13] jFreeDesigner a Java report designer for jFreeReport[11]
[HTTP://www.jfree.org](http://www.jfree.org)
- [14] The Ether Types document available at:
*<http://www.cavebear.com/CaveBear/Ethernet/> or
<ftp://ftp.cavebear.com/pub/Ethernet-codes>*
- [15] RFC 1700 - Assigned Numbers - October 1994
- [16] Linux Socket Programming by Sean Walton published by SAMS

5 Appendix B: Specification