Novel Approaches to the Monitoring of Computer Networks

A thesis submitted in fulfilment of

the requirements for the degree of

MASTER OF SCIENCE

of

RHODES UNIVERSITY

by

Guy Antony Halse

January 2003

Novel Approaches to the Monitoring of Computer Networks

by Guy Antony Halse

Abstract

Traditional network monitoring techniques suffer from a number of limitations. They are usually designed to solve the most general case, and as a result often fall short of expectation. This project sets out to provide the network administrator with a set of alternative tools to solve specific, but common, problems. It uses the network at Rhodes University as a case study and addresses a number of issues that arise on this network.

Four problematic areas are identified within this network: the automatic determination of network topology and layout, the tracking of network growth, the determination of the physical and logical locations of hosts on the network, and the need for intelligent fault reporting systems. These areas are chosen because other network monitoring techniques have failed to adequately address these problems, and because they present problems that are common across a large number of networks. Each area is examined separately and a solution is sought for each of the problems identified.

As a result, a set of tools is developed to solve these problems using a number of novel network monitoring techniques. These tools are designed to be as portable as possible so as not to limit their use to the case study network. Their use within Rhodes, as well as their applicability to other situations is discussed. In all cases, any limitations and shortfalls in the approaches that were employed are examined.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Works of this nature cannot be created in a vacuum, and I am indebted to a number of people for the help and support they have given me throughout the course of this project. I would like to thank everyone who has contributed to the completion of this work, and in particular:

- The Centre of Excellence in Distributed Multimedia at Rhodes University, Grahamstown, South Africa for providing me with the finances and equipment necessary to complete this project. In particular, Alfredo Terzoli, for convincing me to start this project.
- My supervisor, George Wells, for pushing me when I needed to be pushed, and for leaving me to get on with things when I needed to be left to get on with things.
- The systems staff of Rhodes University's Information Technology division, in particular, Kevan Watkins, for providing advice and insight into how the systems at Rhodes function, and for putting up with some of my more "interesting" ideas.
- David Siebörger, a fellow Masters student, for helping me brainstorm, and for pointing out the merits and flaws in some of my ideas.
- Pieter Blaauw for taking me on a guided tour of his workplace, Pick 'n Pay's Information Systems centre in Cape Town, and for giving me some idea of the problems facing the network administrators of large organisations.
- Finally, my parents, for supporting my decision to read for a Masters degree.

— Guy Antony Halse, January 2003

Chapter 1. Introduction

1.1. An Introduction to the Problem Area

The field of network management is of strategic importance to modern computer networks. People are becoming increasingly reliant on computer networks for everyday tasks such as personal and corporate communication, banking, commerce, and shopping. These services are expected to be available at all times, and the cost of failing to meet this expectation is often substantial.

Unfortunately, network management is a field that is fraught with intricacies and problems. One of the most significant of these arises as a result of the way networks have developed over time; the problem of running a hetrogeneous network utilising components from many different manufacturers has frustrated many attempts at providing a unified way of managing and monitoring networks.

Most current network monitoring and management tools suffer from a number of common limitations, and it is these limitations that form the focus of this work.

This project sets out to identify several common problem areas that are not adequately addressed by current network monitoring techniques. It uses the network at Rhodes University as an example of a typical campus network, and notes that it suffers from several shortfalls in terms of its current network monitoring strategy. These shortfalls represent problems that are typical of many other organisations, and as a result, the University forms a useful platform on which to investigate these areas.

Having identified four such problems at the University, this project aims to find novel solutions that provide the University's network administrators with the means to address these issues.

These solutions are then investigated in light of the Open Systems Interconnect network management model to determine how they fit together and form a more complete, and more universally applicable network monitoring solution.

1.2. Document Layout

This document is divided into three logical parts, split across a number of chapters. The first part (Chapter 1 and Chapter 2) contains a definition of the problem, a general introduction of the *status quo* and an overview of the problems associated with this approach. This is followed (Chapter 3 to Chapter 7) by a detailed examination of the problem areas that were identified in the first part as well as the solutions employed to overcome them. The last part (Chapter 8 and Chapter 9) of this work draws these problem areas together to present an overview of the work covered, and draws conclusions on them.

This section is intended to give the reader a brief overview of the structure of this document and the composition of each chapter.

- Chapter 1 contains an introduction to the problem area. It also looks at some of the conventions used in this document and introduces the reader to the Internet standards process and the various Open Systems Interconnect models.
- Chapter 2 uses the Open Standards Interconnect model of network management to identify the goals of network management, and looks at these goals in relation to current network monitoring techniques. It goes on to examine some of the shortfalls of these current methods, paying particular attention to the problem of multi-vendor networks. The chapter concludes with an introduction to the network at Rhodes University, which is used as a case study throughout this document.
- Chapter 3 starts off by looking at simple network monitoring techniques aimed at the networking layman. The problem of multi-vendor networks is revisited and examined in detail. A solution to this issue is proposed, using the extensible markup language (XML) to provide a way of abstracting from the vendor-dependent components. An application that monitors several digital subscriber lines is developed to test this abstraction approach.
- Chapter 4 looks at two different methods of determining topographical information about a network. The first of these, a traceroute(8)-based approach, produces network maps at layer three of the Open Systems Interconnect reference model, while the second, a simple network management protocol-based approach, produces maps at layer two.
- Chapter 5 examines an intrusive way of experimentally determining the growth rate of a particular network. It examines the problems encountered in implementing this approach, as well as some of the social considerations associated with this sort of network monitoring. The problem of representing the vast amount of data gathered by this system is examined, and some unexpected uses for the information are discovered and discussed.
- Chapter 6 provides a means by which network administrators can determine the location, both logical and physical, of a particular host on the network. It explains why this information is important, and discusses a number of problems related to obtaining it. A combined approach that attempts to work out both logical and physical location is developed and discussed in detail.
- Chapter 7 explains that currently most network monitoring tools produce symptom-based fault reports. Several types of artificial intelligence are examined in order to find way of producing more useful fault reports, and an expert system is developed to do this. The result is an “intelligent” method of gathering information about the status of a network and of reporting faults.
- Chapter 8 takes a broad overview of the work covered and explains how each of the systems presented are related to each other, and how they depend on other systems. Each application is examined in relation to the Open Systems Interconnect network management model in order to show how each system relates to the conceptual areas in this model. In addition, the problem of porting each of the systems to other networks is considered, with specific attention to those applications which are not straightforward to port. A comparison to existing solutions is also made.
- Chapter 9 concludes the work and presents some possibilities for future work in this field.

1.3. Conventions and Typography

When writing a document such as this, it is unavoidable that some amount of jargon and acronyms are introduced. In most places, the first instance of a particular piece of jargon or acronym is accompanied by an explanation or definition of the meaning. Where a particular term has not been used for a number of chapters, it may be redefined on its first subsequent occurrence. In addition, there is a *Glossary of Abbreviations* at the end of this document.

Reference is made throughout this document to several Unix™ commands, such as the "traceroute" command. These references are made using the standard Unix man page syntax, the name of the command followed by a number in parenthesis, for example, traceroute(8). The number in parenthesis indicates which section of the manual documents the command in question. Sections that will typically be seen in this work are sections one, five and eight. Section one documents general commands, utilities and command line programs, section five documents system file formats and section eight documents system maintenence commands. In all instances, the manual pages (colloquially known as "man pages") referred to are those of the FreeBSD operating system. These man pages can be retrieved via the Internet from the FreeBSD web site (`http://www.freebsd.org/cgi/man.cgi`).

A number of references are also made to RFC (Request for Comments) and BCP (Best Current Practice). These refer to documents that form part of the Internet standards process, which is explained in Section 1.4.

1.4. Internet Standards Process

This document makes reference to a significant number of "RFC"s, or Request(s) for Comments. These RFCs form part of a set of technical documents and organisational notes that describe the structure of the Internet. RFCs are published by the Internet Engineering Task Force, and are written by various Internet standards bodies, and the Internet community as a whole.

The criteria for the publication of RFCs is fairly straightforward. All RFCs begin their life as an Internet draft. These Internet drafts are works in progress and have a life-cycle of six months, after which time they are either removed, revised, or become an RFC. Internet drafts are available for informal review and comment. Internet drafts may only become RFCs if they are judged to be sufficiently mature and of interest to the Internet community.

An RFC may be published either as a "standards track" document, or an "Informational" or "Experimental" document. Standards track RFCs document the workings of the Internet, and the various protocols which make it up. Experimental and Informational RFCs do not describe standards, but rather contain information that may be of interest or use to the Internet community. These non-standards RFCs often indicate the best current practices in relation to a specific area of the Internet. These best current practices (BCP) form a separate document series.

All standards track RFCs are subject to a strict process of review before they are adopted as Internet standards. Every standards track RFC has a status associated with it which indicates its maturity and place within the standards track. A standards track RFC starts out as a “Proposed Standard”, after which it becomes a “Draft Standard” before being confirmed as an “Internet Standard”.

Internet Standards are specifications for which a significant amount of review, implementation, and operational experience has been gained. They have a significant amount of technical maturity and are generally of interest to the Internet community as a whole.

All RFCs are identified by a unique number, starting at one and incrementing sequentially as documents are published. This number is assigned to a specific version of the specification and after an RFC has been assigned a number it may not be edited. Subsequent revisions start out as Internet drafts, and are assigned a separate RFC number when they are ratified.

A repository of RFCs is maintained by the RFC editor (`http://www.rfc-editor.org/`) and this collection is mirrored on servers world-wide.

The Internet standards process is described by [BCP 9].

1.5. OSI and the ISO

The Open Standards Interconnect (OSI) series of models are an attempt by the International Standards Organisation (ISO) to standardise the way that computer systems communicate with each other. Although there are several OSI models, the most widely used one is the OSI reference model. This seven layer model is intended to ensure interoperability between different protocols and methods of communication. The layers of the OSI reference model are: physical, data link, network, transport, session, presentation, and application [Tanenbaum, 1988].

- The physical layer (or layer one) is concerned with the transmission of raw binary data over a communications channel. Protocols at this layer define, for example how each side of a communication sends and receives ones and zeros.
- The data link layer (or layer two) takes the raw transmission function and converts in into an error free transmission channel. It merely ensures that the stream remains complete, without any regard to structure or content. Most switching devices operate at this layer, as does the Address Resolution Protocol (ARP).
- The network layer (layer three) concerns itself with the routing of packets from source to destination and the basic operations of a subnet. Most routers are layer three devices, and the Internet Protocol (IP) resides at layer three.
- The transport layer (layer four) has the task of accepting data from the session layer, breaking it into smaller parts if necessary and passing it to the network layer. This layer also reassembles the parts at

the destination and ensures that all parts are correctly received. Protocol based switching operates at this level, as does the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

- The session layer (layer five) allows users on differing machines to establish sessions between the machines. In addition to ordinary data transfer, the session layer provides enhanced services, such as authentication. This layer is not present in most Internet protocols.
- The presentation layer (layer six) ensures that data is encoded in the correct format for the application or transport that is being used. If necessary, this layer performs the necessary translations between different encodings. The IP security protocol, IPSEC, may be an example of the application of layer six.
- The application layer (layer seven) is where all end-user applications sit. This layer supports the hundreds of different user protocols used to perform various tasks, such as e-mail, file transfer, *et cetera*.

This work also makes reference to another, less commonly used, OSI model — the OSI network management model. This model describes the tasks associated with managing modern computer networks, and provides a way to define relationships between various tasks. The network management model is described in Section 2.1.

Chapter 2. Overview Of Network Monitoring

It is important to understand some of the goals of network management before the problems associated with current network management techniques can be examined. Once these techniques of network monitoring, and the problems associated with them, are understood the more specific goals of this project can be examined. These goals need to be put in context, however, and this requires some knowledge of the network infrastructure and layout at Rhodes University.

2.1. Goals of network monitoring

Modern computer networks tend to be large heterogeneous collections of computers, switches, routers and a large assortment of other devices. To a large degree, the growth of such networks is *ad-hoc* and based on the current and perceived future needs of the users. As networks get larger and faster, the job of monitoring and managing them gets more complex. However, the job of managing computer networks becomes increasingly more important as society becomes more dependent on computers and the Internet for every day business tasks. Network downtime now costs significant amounts of money [CPR, 2001] so it is important that network and system managers are aware of everything that is happening on the networks for which they are responsible. Fortunately, computers are fairly good at watching other computers which means we can automate this task to some extent.

In their discussion on the basics of network management, Cisco Systems point out that the term “network management” means different things to different people [Cisco, 2002]. They give two examples at opposite ends of the spectrum to illustrate this diversity: A solitary network consultant monitoring network activity and high end workstations generating graphical views of network topologies and traffic. Both of these examples employ some form of tool to gather, analyse and represent information about a computer network; therefore, in general, network management involves a set of tools to aid people to monitor and maintain computer networks.

In an attempt to better understand the goals of network monitoring, it is useful to have a model of some kind.

The International Telecommunications Union (ITU) proposed a network management model aimed at understanding the major functions of network management and monitoring software. This management model forms part of the X.700 series of documents from the ITU and is based on the Open Systems Interconnect (OSI) reference model. It is in the process of being standardised by the International Standards Organisation (ISO). It addresses five conceptual areas, being: performance management, configuration management, accounting management, fault management and security management [Rose, 1991].

These conceptual areas are useful in understanding the goals of network monitoring and management, but first there is need to differentiate between the two. The difference between network management and networking monitoring is blurred — people tend to use the two terms interchangeably. For the purposes of this document the term "monitoring" will be used to refer to systems that simply observe and report on a network, without taking any corrective action of their own accord. The term "management" will be used to refer to systems that both monitor a network and take corrective or preventative maintenance action without the need for intervention. As such, "network monitoring" is a subset of "network management". For this reason, although the ISO model refers to network management, a large proportion of the ideas it contains are applicable to the role of network monitoring. The five areas contained in the ISO model will now be examined in more detail:

Fault management is the detection of problems and faults on the network. Such faults should be properly logged, and if appropriate an alarm should be raised. This area is responsible for proper problem identification, determining the cause of the fault and ensuring the proper resolution of the problem. Management software operating in this area may attempt to correct faults on its own, whereas monitoring software relies on notifying somebody of the problem so that they can intervene.

The aim of *configuration management* is to keep track of the network's configuration, both hardware and software. This area includes keeping track of what computers and networking infrastructure are on a network, and how they are interconnected. In addition, configuration management includes following what software versions each device is running, as well as the software configuration of each device.

Security management incorporates all aspects of authentication and access control, from the definition of access policies to the enforcing of those policies. Security management software may need to be aware of access control lists (ACLs), users' access levels, and all other areas of security policy. All transactions should be properly logged to create an audit trail. Exception reports can be generated for events that fall out of the scope of the defined policies, and these reports can be used to alert administrators of the policy violation.

The area of *performance management* looks at the current and expected performance of the network. Elements of network performance that may be monitored include availability, response time, error rate, throughput and utilisation. This information may be compared to theoretical performance levels, historical averages or norms in order to determine how well the network is currently performing. Erratic behaviour and unusual changes in performance may help to predict network faults before they occur, enabling network managers to take preemptive measures. Historical performance information of this sort may be used to determine network growth and predict usage patterns. This data can, in turn, be used to aid in network capacity planning.

Accounting management covers two broad areas: asset control and cost management. Asset control refers to knowing what computers are on the network, who they belong to, who is using them, and perhaps where they are located. The second area, cost management, looks at what the costs of providing network services are and how they are paid for. This may include charging models that see users in some way pay for the resources they use. The management of such charging models, as well as the gathering of any data required to implement them falls within the scope of this area.

In practice there is often a large amount of operational overlap between these five “network management” areas.

2.2. Traditional approaches

2.2.1. SNMP

One of the most widely used approach to network monitoring and management is the Simple Network Management Protocol (SNMP). This protocol was originally formulated in 1988 through RFC 1067. Since then it has undergone many changes and is currently in version three of the protocol (as defined by RFC 1157). SNMP has two distinct parts, a format for describing data about computers and a protocol for transmitting that data over a network.

The format of SNMP data takes the form of a tree structure, with each node in the tree being identified by a numerical Object IDentifier (OID). For example, `.1.3.6.1.2.1.1.1.0` contains a string that describes the system being queried. The idea of OIDs significantly predates the introduction of SNMP, and was standardised by the ISO. The OIDs used by SNMP are a subset of this ISO standard, which means that all SNMP OIDs are prefixed by `.1.3.6.1.2`.

Like Internet Protocol addresses, these OIDs are difficult to remember. For this reason, a method of assigning more human-readable names to these OIDs has been defined. This method uses a Management Information Base (MIB) to convert the numeric OIDs into strings, much like DNS domain names. Using this system, the example OID given above would become `.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0`. To a large extent, the information in these MIBs is standardised — but not always in a useful way as shall be seen in Section 2.3. The MIB used by SNMP is known as MIB-II and is defined in RFC 1213.

The SNMP network protocol itself is fairly simple; it provides several operations to allow a monitoring station to read and write data to a remote host. Operations to read data include `get-request` (to retrieve a specific OID), `get-next-request` (to retrieve the next OID in sequence), and `get-bulk-request` (to retrieve a block of related OIDs). Data can be written using a `set-request`. In addition, SNMP defines a way for the remote host to notify a monitoring station of an extraordinary event that has occurred (OSI fault management). These events are sent as a `trap`, and are a one-way communication between the two hosts.

2.2.2. RMON

RMON is a powerful tool for the remote monitoring of Ethernet networks. At its heart is a standard SNMP MIB, defined by RFC 2819. RMON agents on remote network devices gather information about

the system they are monitoring and make it available over the network using the RMON MIB. This information can then be collected and processed by remote management stations.

The idea behind RMON is to provide a standard method for monitoring the basic operation of Ethernet networks, and to provide interoperability between SNMP management stations and remote devices. RMON agents on these remote devices provide a powerful alarm and event notification mechanism for alerting the management station of changes in network behaviour. These alerts are sent using the SNMP `trap` method.

RMON agents have the ability to automatically collect and store historical information in order to provide trend data on such basic statistics as utilisation, collisions, *et cetera*. Network management tools can retrieve these histories and analyse them in order to understand network usage patterns (OSI performance management).

In total, RMON specifies ten services. Nine of these RMON groups (as they are known) apply to Ethernet and one is specific to token ring networks. Not all RMON agents will implement all the RMON groups, since some of them have extensive processing or memory overheads associated with them. These RMON groups include: general statistics about the network, long term histories, alarm events, host-specific statistics, packet and error counts for each conversation, packet filtering, packet capture, and event logs.

RMON agents have the ability to collect network traffic at the Media Access Control (MAC) level based on a defined packet filter. These packets can be retrieved from the agent and processed by a protocol analyser, providing a way to remotely monitor the traffic on a network segment in much the same way as tcpdump(1) does on the local segment.

2.2.3. Active probes

Both RMON and tools using SNMP fall under the heading of active monitoring; that is, they actively attempt to retrieve information from remote hosts. Dedicated management protocols, however, are not the only form of active network monitoring. Useful information about a network can often be gained from querying remote hosts using normal communication protocols. The reachability of a particular device can be tested using tools like ping(8), for example.

It is often informative to test a particular network service by using that service directly. To check that a web server is functioning correctly, for example, one could connect to the http port (port 80) of the host that runs the web server and use the HyperText Transfer Protocol (HTTP) to request a web page from that server. If the response that the server gives is correct, the web server may be presumed to be functioning normally.

These sorts of active probes often provide the most accurate information about the state of network services, since they directly test the service in the way it gets used.

2.2.4. Passive monitoring

Many network monitoring tools are designed to passively watch network traffic on a particular subnet or passing through a particular gateway. By examining all the packets as they go past, one can often learn a lot about the way a network is running. For example, arpwatch(8) looks for Ethernet Address Resolution Protocol (ARP) "whois" messages on the network and compares the responses to those it has stored in a database. In this way, arpwatch(8) can tell when a new device is plugged into the network or when a device changes its IP address or MAC address. This information is often useful in controlling who uses the network (OSI security management) or for troubleshooting faults such as IP address conflicts (OSI fault management).

Passive network monitoring is often the simplest form of monitoring to implement, since it does not require any cooperation from the monitored hosts. Since it looks directly at the traffic passing over the network, this form of monitoring has a huge potential to provide useful information about the state of network, both past (through the use of logging) and present.

2.2.5. Proprietary protocols

Many vendors of network infrastructure define their own management protocols. In general, these protocols are specific to that manufacturer's products and are often used as a source of competitive advantage.

A good example of this is the Cisco Discovery Protocol (CDP), which is used by Cisco routers and switches to discover their neighbours. Each CDP compatible device listens on a well-known multicast address, and periodically announces itself on that address. The table of known neighbours can be retrieved from a CDP compliant device by any network management product using SNMP [Cisco, 2001]. This capability is used by CiscoWorks (Cisco's commercial network management tool) to manage clusters of switches.

2.2.6. Higher level products

So far fairly low level network monitoring applications and protocols have been examined. Many applications build on these protocols to provide a higher level view of the network. Most often, these programs attempt to represent various aspects of the network in a graphical format.

Perhaps the most popular open-source network monitoring package is Tobias Oetiker's MRTG, the Multi-Router Traffic Grapher (`http://www.mrtg.org/`). MRTG uses SNMP to gather information about various network devices and displays this information graphically on a web page. Historical information is stored in a Round Robin Database (RRD) and this data is used to provide trend information in the graphs [Oetiker, 1998].

MRTG is an example of the most common visual method for representing network information, that is a variable versus time graph. The variable can be anything from router load to the amount of traffic on an interface. Many commercial packages also use this approach as their primary method for displaying information that they have gathered. A good example of this is the Optivity package from Nortel Networks [Nortel, 2002]. This form of monitoring is the performance monitoring mentioned in the OSI model.

Big Brother is a web-based network monitoring tool that is free for non-commercial use [BigBrother, 2002]. It actively probes hosts in an attempt to determine whether everything is running correctly. In its most basic form, Big Brother simply checks on the reachability of devices. It understands a limited subset of network protocols and can be configured to check the status for services such as a web server. Configuration is done through text-based configuration files, with no facilities for automatically detecting the presence of new hosts. It has limited abilities to provide real-time notification through the use of pagers. Big Brother keeps a historical record of status changes which can be viewed through its web interface. Big Brother fulfils the fault management, and to a lesser extent the performance management, role of the OSI model.

This is an example of a common method for representing real-time system status. It provides the network manager with an up-to-date, real time view of the status of important network services. Big Brother uses the colour of the background to indicate the state of the most critical problem on the networks, thus allowing the network manager to tell, at a glance across a room, if the system is functioning correctly. This is a important feature in today's fast paced world.

There are certainly many other high level approaches to the problem of monitoring networks, but these three examples illustrate two of the more widely used methods for representing network information. These approaches are employed later in this work.

2.3. Problems associated with traditional approaches

2.3.1. Limitations of current methods

Although widely used, SNMP is not without its limitations [CMU, 2000]. The most widely implemented version of SNMP is SNMPv1, which lacks any form of authentication, relying instead on "security by obscurity". This limitation has caused many vendors to omit parts of the SNMP specification from their implementations in order to prevent access to critical network devices.

Wellens points out that SNMP is not particularly suited to the problem of fault management, but rather to that of performance monitoring — "Nearly 100% of network management occurs when networks

are not failing." [Wellens, 1996] The article goes on to say that the value of SNMP as a tool for fault finding is somewhat below that of traceroute(8), ping(8), and nslookup(8). Unfortunately, there are very few, if any, network monitoring tools that utilise these simple building blocks.

Wellens also explains that the MIB definitions that have been produced for SNMP are SNMP's most valuable legacy [Wellens, 1996]. They comprise the collective thoughts of many experts on what exactly constitutes valuable data for network monitoring. These MIBs, however, are not without their own limitations, as shall be seen in Section 2.3.2.

SNMP is a polled system. It requires than monitoring stations actively poll devices for information. This can result in significantly increased network traffic if the polling frequency is not carefully monitored. Since SNMPv1 provides no inherent security, this can be difficult to achieve.

RMON was originally developed to overcome some of the limitations in SNMP, most notably the fact that SNMP agents can only store very limited amounts of information. This ability to store information reduces the necessary polling frequency which in turn should reduce the network load.

Since RMON is largely based on SNMP, it too will suffer from some of the limitations that have already discussed. RMON is not without limitations of its own though; not all RMON agents implement all parts of the protocol, so it is not always possible to retrieve the information one requires from a device using RMON. Using RMON requires a good knowledge of the network topology in order to place agents at the best possible positions to gather information.

Being able to successfully use methods such as SNMP and RMON requires that one has a good network monitoring package. These packages are usually produced by network vendors and are designed specifically to meet the needs of their devices. This problem of vendor-specific solutions is discussed in Section 2.3.2.

Network monitoring is of little use without accurate, up-to-date reports that allow network managers to make correct decisions. Unfortunately, fault reporting seems to be one of the field's weakest areas. Traditionally, fault reports are limited to the symptomatic reporting of the problem, with the network manager being left to interpret these symptoms — much like a doctor diagnoses ailments from a list of symptoms. This method works very well in a large number of cases, but breaks down in loosely structured organisations where the realm of responsibility is not delegated to one body. For example, being told that the departmental mail server has stopped working is not particularly useful if the underlying ailment is a faulty DNS server managed by a different organisational division. In cases like these, network monitoring software should be able differentiate between faults that are within the divisional realm of responsibility, and those that fall beyond it. This is something that symptomatic reporting does not cater for very well.

2.3.2. The problem of multi-vendor networks

Modern computer networks require that thousands of different devices from hundreds of different man-

ufacturers successfully communicate with each other. For this reason, standards have been developed to ensure that products from different manufacturers interoperate successfully. In the network monitoring world, standards such as MIB-II have been designed to provide an easy way for network management products to interoperate with the plethora of devices on the market. Unfortunately, technology often evolves faster than the standards that cater for it.

The SNMP MIB-II caters for this eventuality by providing the `enterprises` object identifier, where manufacturers can create their own, device specific, MIB trees. As technology advances, the gap between the standardised MIBs and the requirements of the technology increases. This results in more and more information about a device being stored under the `enterprises` OID, to the point where, when dealing with modern technologies such as Rate Adaptive Digital Subscriber Lines (RADSL), a vast majority of the management information about the device is only available through the `enterprises` OID.

Since the `enterprises` OID is not, by necessity, standardised, one ends up with a situation where products in the same class of device (e.g. router) from different manufacturers, either by coincidence or by choice, use different OIDs to represent the same information.

Network management tools using these `enterprises` MIBs are, in general, limited to working with devices from one vendor. The effect of this limitation is that, in order to fully manage a network using these tools, organisations are required to implement a homogeneous network — that is, a network where all network infrastructure and monitoring tools are sourced from a single vendor.

Unfortunately this is not always desirable or possible. Networks tend to grow in an *ad-hoc* fashion, depending on the demands of the users of that network. This is especially true of small organisations that do not have the resources to implement long term strategic plans. The result is usually a hetrogeneous collection of devices from different manufacturers.

Management of multi-vendor networks using SNMP has historically always been problematic. This divergence was the reason that MIB-II was initially standardised [RFC 1213]. The problem can only get worse as network technologies diverge more and more from the “norms” that underly MIB-II.

This problem is compounded by the use of proprietary protocols as already described in Section 2.2.5.

2.4. Network monitoring at Rhodes University

2.4.1. Overview of the network at Rhodes University

Rhodes University (referred to subsequently as “Rhodes”) operates a network consisting of approximately 1400 computers connected together to form a campus-wide local area network (LAN). The network encompasses almost all of the Grahamstown campus and extends to the University’s office

in Johannesburg and the satellite campus in East London, as well as to various parts of Grahamstown through remote access technologies. The University backbone runs on 1Gb/s fibre, and the majority of machines access the network at 100Mb/s. Satellite offices are connected using Diginet leased lines, and access technologies such as ISDN dial-up, RADSL, analogue leased lines and dial-up are also employed to a limited extent.

For ease of management, the Rhodes LAN is divided up into about 32 subnets, each of which serves a different logical area of campus (and usually a different physical area too). For many years, these sub-divisions were sufficient to distinguish different areas of the network. In recent years, however, it has become necessary to further sub-divide these subnets to allow machines with different logical functions to be grouped together. In general these secondary divisions are not subnets in their own right, but are placed on CIDR boundaries to make it easy to refer to entire logical blocks of computers.

In general, networking is managed by the Information Technology Division, a support division falling under central management. In rare cases there are exceptions to this rule, most notably, the Department of Computer Science. This department manages its own, fairly extensive, network infrastructure in cooperation with the Information Technology Division.

2.4.2. Monitoring

There are three network monitoring approaches taken by Rhodes.

The first of these is the use of commercial network monitoring packages; namely CiscoWorks from Cisco and Optivity from Nortel Networks. Both these products are capable of monitoring large numbers of devices on the network, and this was the original intention. In practice it was found that these products did not provide a satisfactory solution to monitoring Rhodes' network, and currently each monitor a single device on the network — a Catalyst 5000 switch in the case of CiscoWorks and a Passport 8600 switch in the case of Optivity. These two devices form the core of Rhodes' network, and so are fundamental to the running of the network.

Network services, such as e-mail, the University web server, the central file servers, *et cetera*. are monitored by the Big Brother package mentioned earlier. This package is also used to monitor the South East Academic Libraries Service (SEALS) that is hosted at Rhodes. Big Brother is connected to a SMS notification service to provide rudimentary notification of problems as they occur. It is also used to keep historical information about the reachability of various machines, something that is important in terms of the service level agreement between SEALS and Rhodes.

TENET, the University's service provider, uses Tobias Oetiker's MRTG to monitor each of the down-stream links they provide. They make these graphs available to the University and these are used to determine both network faults and historical performance. The graphs clearly indicate the Committed Information Rate (CIR) and allow the University to monitor and manage their bandwidth usage.

In addition to these, the University has made use of some custom-written network monitoring software in order to determine specific statistics, such as the amount of traffic on each of the various subnets. Due to time constraints, this software has not been well maintained and often does not reflect the current state of the network.

2.4.3. Monitoring shortfalls

Several shortfalls in the current monitoring systems at Rhodes have been identified, and later chapters will attempt to address some of these.

To a large extent, access control for the University's proxy server and firewall is dependant on hosts having a valid reverse DNS entry. DNS records are maintained in a centralised database together with information about the host (its MAC address, asset number, owner, *et cetera*). Unfortunately, there is no easy way of telling when records in this database become stale. The result of this is that there are in excess of 5300 DNS entries at Rhodes corresponding to about 1400 actual hosts (or roughly 3.5 times as many). This makes bypassing DNS-based access control fairly trivial.

The University has no real way of determining which of these records are still in active use, or indeed how many actual machines there are on campus. There are no statistics about which subnets are most populated, or about the growth rate of the University network. All present information about these metrics is currently of the *educated guess* form.

Over the years, the University's network has grown in a largely *ad-hoc* fashion. As a result, there are no complete topology maps of the network. This is slowly being addressed, but it is being done manually — every time a section of the network is worked upon, its layout is documented. Attempts at automatically working out the topology of the network have failed, largely due to the multi-vendor problem discussed in Section 2.3.2.

In reality, however, this documentation is not always kept as up-to-date as it is supposed to be. When the network infrastructure in Hamilton Building (home of the Department of Computer Science) was put in, detailed records of all connections were made. These records, however, have been misplaced and currently there is no way of associating a physical network point in the building with a corresponding port on the local switch stack. This problem is not unique to the building, or indeed even to Rhodes. For example, Pick 'n Pay have the same problem in their offices in Rondebosch, Cape Town.

2.5. Summary

This chapter provides the groundwork and background information for future chapters. It presented an overview of the field of network monitoring, using the Open Systems Interconnect model of network

management to provide a structure for this discussion. The terms network monitoring and network management were defined in relation to this model.

Current techniques of network monitoring are examined, and the limitations of such methods are introduced. In particular, the problems associated with multi-vendor networks are discussed, as are the related problems associated with the Simple Network Management Protocol.

An overview of the network at Rhodes University is provided, and some of the problems with the current techniques of network monitoring employed on this network are investigated. These problems form the basis of this project.

Chapter 3. First Steps

In order to become familiar with current techniques of network management and get a feel for the *status quo* a number of simple problems were investigated. The limitations of current solutions were explored and new solutions evolved in an attempt to get around these limitations. This represents the first steps towards a more complete solution for monitoring networks.

3.1. Solutions for the layman

As computer networks become more widely used, situations often develop where those people tasked with the day-to-day running of the network are not sufficiently skilled to correctly diagnose problems when they occur. Often these networks are remotely managed in such a way that the people on-site rarely need to intervene.

Unfortunately, when intervention is necessary, it is usually because the network is inaccessible to those tasked with managing it remotely. These remote mangers need to be able to make an accurate diagnosis of a problem they cannot see, relying solely on information that is provided to them by the site. However, it is often difficult and time-consuming to get an accurate problem report from someone who does not understand why the problem is occurring.

To make the diagnosis of remote network faults easier, a number of simple tools can be provided. These tools should attempt to determine the cause of the problem as accurately as possible (Chapter 7 discusses this in detail) and provide a list of possible causes and solutions. In cases where solutions are beyond the capabilities of those people who are on-site, the tools should provide sufficient information for diagnosis to be made, for example, telephonically.

A fault management system meeting these criteria was developed for a local high school. This system consists of a single web page which is served off their local server. The page sequentially tests the reachability of every host and router that the school depends on for their upstream Internet connectivity.

Results are presented in a large table in which each row corresponds to a test. Those systems that are functioning correctly are indicated by large blocks of green, while those that fail the reachability test are indicated by large blocks of red. The tests are ordered in such a way that the first red block on the page is the most likely cause of the error.

Beside each block is the name of the host or router that is being tested, a description of the function of that host in the provisioning of connectivity, and a detailed description of the action to be taken if the test fails. Where a test is made on a host that the school has no direct control over, for example, an upstream proxy server, information is given on how to go about reporting the fault.

The information on this web application is presented in such a way that it is easy for someone to read it over a telephone, enabling the remote administrator to virtually perform the same tests that would

normally be required to diagnose the fault. Thus, this tool allows the layman to diagnose and rectify simple faults on a network, and to provide a remote network administrator with sufficient information to resolve those problems that are beyond those people who are on site.

3.2. Multi-vendor networks revisited

Section 2.3.2 looked at some of the problems associated with multi-vendor networks. Specifically, it looked at SNMP, and how the standard MIBs do not adequately cater for network technology.

One of the biggest hindrances to MIB-II is the standards process itself. All revisions to the MIB-II RFC require that it follow the full standards, from Internet Draft through to acceptance as an RFC [BCP 9]. This can take a significant amount of time. Further to this, there are usually considerable delays between the publishing of a new standard and vendors implementing it. The result is that it is almost impossible for MIB-II to keep up with the advances in technology. In fact, RFC 1213 has not been updated since November 1996.

For example, MIB-II assumes that all internet traffic is symmetric — that is that the media has the same amount of bandwidth in both directions. As such, it only has one entry per interface in the MIB to record the speed at which the line is running, namely `interfaces.ifTable.ifEntry.ifSpeed`. This assumption was valid in 1991 when RFC 1213 was written, and in 1996 when it was updated by RFC 2011. Recent developments, such as ADSL and the ITU's v.90 standard for 56K analogue modems, have changed this. Any modern technology that uses asymmetric transfer rates is not catered for by MIB-II. This forces manufacturers to include the extra information in the `enterprises` section of the MIB, usually in a proprietary way.

These inconsistencies make writing vendor-independent network monitoring software that caters for these new technologies an onerous task. Such software needs to be aware of what OID each vendor uses for particular bits of information. There are two ways of dealing with this: either one hardwires the values into the program (resulting in the support of a limited subset of vendors), or one provides some way for the software to work out what OIDs it should be referring to. Since the first of these approaches is clearly not ideal, these focus will be on the second.

What is needed is a way to consolidate the various vendors' `enterprises` entries into a single format that is understood by the network monitoring tools.

In its simplest form, such an amalgamation method could be a text-based configuration file that is read by the monitoring software. It would be more useful, however, if this information could be shared between applications. The ability to do this requires a layer of abstraction between the devices being monitored and the network monitoring applications.

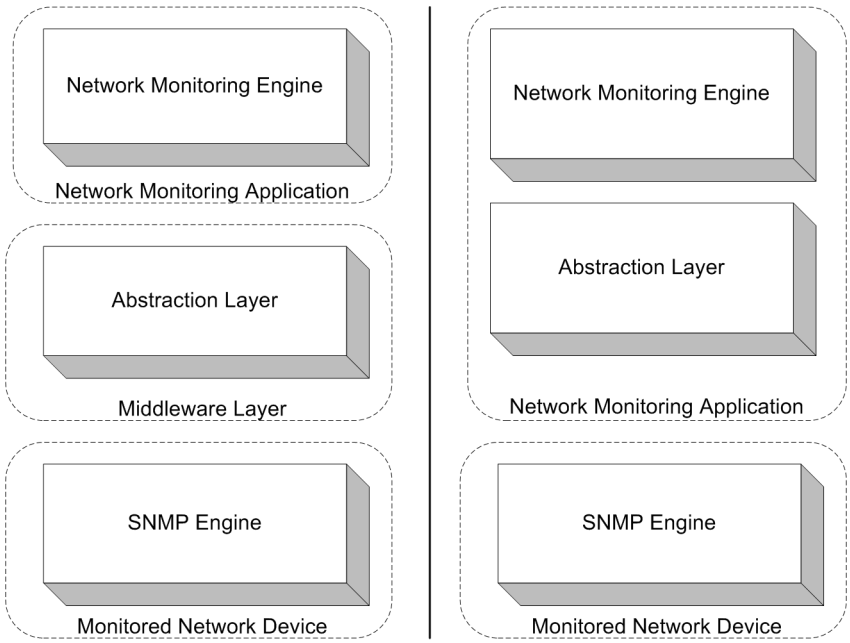
There are two ways of achieving this abstraction: by creating a middleware application that translates requests for information from the monitoring software into SNMP requests to the network hardware being monitored, or by creating a method for configuring the monitoring program that is both abstract

and flexible enough to be shared between different programs. These two approaches are shown in Figure 3-1, with the middleware approach on the left. Both these approaches use essentially the same idea, the first simply formalises the idea of a middleware layer by making it a separate entity.

The problem with having a separate middleware layer is identical to the one faced by MIB-II. The middleware application would have to keep abreast of advances in technology changes, and vendors would be relying on a third party to do this. In all likelihood, the middleware application would quickly become out of date, leaving users no better off than they currently are.

For this reason, having the abstraction layer embedded within the monitoring application itself seems to be the most feasible approach. The problem then lies in making the abstraction information easily portable between applications. This requires that the format that the abstraction information is stored in be standardised, or at least formalised to some extent.

Figure 3-1. The two abstraction methods



The eXtensible Markup Language (XML) is a *meta language* that allows one to define new markup languages [W3C, 2003]. A markup language can be developed using XML that would allow us to pass the consolidation information around in a standardised way. If done correctly, this markup language would enable vendors to rapidly share information about new products.

Work was done on developing a simple proof-of-concept version of such a language. Several Document Type Definitions (DTDs) were developed in an attempt to accurately model both the structure of the network and the various network devices that make up the network. An early example of such a DTD

is given in Figure 3-2.

Figure 3-2. Example XML DTD for network monitoring

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE monitor [
  <!ELEMENT monitor (host+) >
  <!ELEMENT host (ports?, oids?) >
  <!ELEMENT ports (port+) >
  <!ELEMENT port (#PCDATA) >
  <!ELEMENT oids (oid+) >
  <!ELEMENT oid (#PCDATA) >
  <!ATTLIST host name CDATA #REQUIRED >
  <!ATTLIST host port CDATA 161 >
  <!ATTLIST host community CDATA public >
  <!ATTLIST oid name CDATA #REQUIRED >
  <!ATTLIST oid type (text|gauge|counter) text>
  <!ATTLIST oid precision (16|32|64) 32 >
]>
```

This DTD attempts to describe both the layout of a local area network as well as the methods for communicating with each device on the network. In many ways, this DTD is both overly complex and overly simple. It is overly complex because it tries to describe two things at once: a communications protocol and a network layout. As such, it does not differentiate between information specific to the network layout and information specific to a particular class of network devices. For example, all Cisco 2900 switches store their system description in the OID `.1.3.6.1.2.1.1.1.0`, but not all Cisco 2900 switches have the same hostname. These two different classes of information should be separated, perhaps into two distinct DTDs. This separation was attempted in later revisions of the DTD.

The DTD is overly simple because it does not yet allow for the complete description of all network devices. It is not abstract enough to be able to handle the rapid changes in network technology, nor is it refined enough to be applicable to all existing technologies. As a proof-of-concept, however, it was sufficient to allow some testing to be done. Two network monitoring applications were built upon this DTD in order to determine whether the idea of an abstraction layer at this level was feasible.

The first of these was a monitoring application for the Computer Science Department's remote access subnet, and is described in Section 3.3. The second was an application to do automatic network topology discovery on Rhodes' LAN, and is described in Chapter 4. Both these applications showed that, while using XML to create an abstraction layer was feasible, a lot more work would need to be done on this approach to make it universally applicable.

For an XML-based approach such as this to work, it would need active community support. This means that the DTD would have to be publicly available and open to public discussion. Without such peer review, it would be impossible to develop a model that was generic enough to have wide scale use, since it is impossible for one person to anticipate every possible network layout and configuration.

There may be some merit in reworking this DTD to be an XML schema, since there is a trend towards the use of schemas [W3C, 2000]. This would make the system more universally applicable, since it would be in keeping with an accepted way of doing things. For the purposes of this project, however, the representation as a DTD is sufficient to allow other applications to be built using this concept. This is especially true since the applications developed using this approach use the Expat (`http://expat.sourceforge.net/`) XML parser, which is a non-validating parser.

3.3. RADSL monitoring

As part of its research equipment, the Department has several Rate Adaptive DSL Access Multiplexers (DSLAMs). These DSLAMs provide remote access to several members of staff and senior students. The notable thing about these, from a network monitoring perspective, is that they run the line with asymmetric bandwidth. The upstream RADSL link runs at anything up to 1Mb/s and the downstream bandwidth is anything up to 7Mb/s. It is possible for the bandwidth in each direction to be the same, but in the general case it is not.

Section 3.2 discussed the implications of such asymmetric bandwidth on normal SNMP monitoring techniques. For this reason, a custom application had to be written to monitor these lines. As was mentioned in Section 3.2, this application was written to test the feasibility of using an XML-based abstraction layer.

Since its inception, this monitoring tool has undergone many revisions to increase its functionality and make it more useful. Several problems were encountered in its development, some of which took significant amounts of time to resolve.

The most significant of these was locating appropriate MIBs and documentation for the DSLAMs. The Department has two different models of DSLAM. The older model uses RADSL modems at the remote end, and all management information for these devices is stored on the DSLAM itself. The newer model uses routers at the remote end, and these routers have built in SNMP agents to handle requests for information about them. Unfortunately the manufacturer refers to the SNMP agents as “protected”, meaning that they are only accessible through the interface provided by the DSLAM. The documentation on these “protected” agents is somewhat poor, and it took many months and several e-mail messages to the the manufacturer in order to figure out how to address them. In the end it was almost purely by accident that the correct way of talking to them was discovered. While browsing an SNMP walk of the DSLAM with a new MIB, an OID called `entityMIB.entityMIBObjects.entityLogical.entLogicalTable.entLogicalEntry.entLogicalCommunity.1001000` was noticed. This OID had a value of `public@s1p1`. When the DSLAM was queried with this as a community name, it produced data from the DSL modem connected to slot one, port one.

The application, which fulfils the *performance monitoring* section of the OSI model, consists of two parts. The first is a data collector and the second is a graphical user interface that is used to display the

data.

The data collection application is called by cron(8) every five minutes and polls each of the DSLAMs in order to obtain data about each of the RADSL lines attached to it. Data is collected on the up and downstream line speeds, the amount of traffic on the line, and the error and discard rates associated with the line. This collection is done using SNMP to each DSLAM, and in the case of RADSL routers, to each router as well.

As SNMP uses UDP as its transport protocol, care had to be taken to ensure that each of the devices the application wished to query was reachable. In the case of the two DSLAMs, it can assumed that this is the case since they are always switched on; this assumption does not hold true for remote routers, which are turned on and off at the whim of the user. The penalty for trying to connect to a router that is not currently on is that the application has to wait for the full UDP timeout (five seconds) before it can decide that a particular router is down. For this reason, the data collection application is careful to query the controlling DSLAM for its list of active interfaces, which it uses to determine which routers it should attempt to poll. This problem would have been compounded if SNMP had chosen to use TCP since the default TCP timeout is significantly longer (about 75 seconds on a FreeBSD machine). Although both these timeouts are configurable, it rarely makes sense to do so — the defaults are chosen to give best performance in most scenarios.

The information collected from the remote SNMP agents is processed to ensure that it is valid — that, for example, data on the line speed never exceeds the theoretical maximum for the line. Any erroneous data is discarded (actually, its recorded as an “unknown” value) and the validated information is stored in a round robin database using rrdtool(1) (`http://www.rrdtool.org/`).

By design, RRDs consolidate data over time. The consolidation function is applied to the average, maximum, and minimum values creating three separated groups of data. The RRD used for this application is configured to keep 600 5–minute samples (50 hours), 700 30–minute samples (14 days), 775 2–hour samples (58 days), and 797 1–day samples (just over two years).

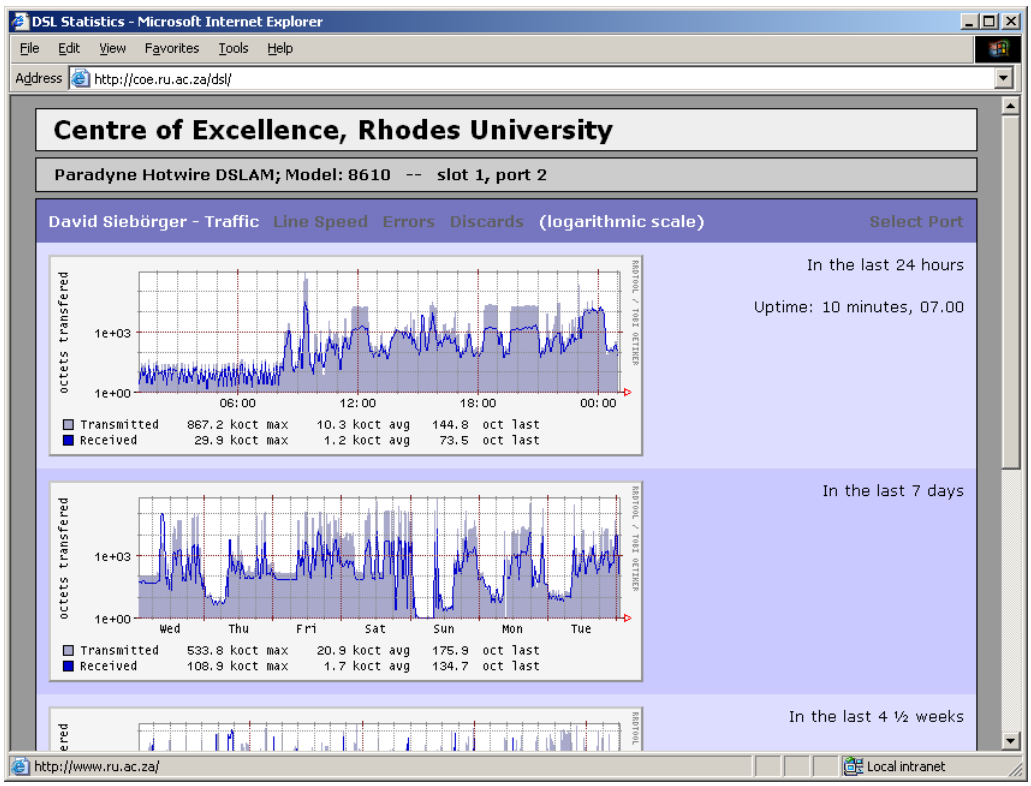
This configuration was chosen partly because it gives us roughly an equal number of samples over each of a day, a week, a month and a year. These periods are used for graphing as shall be seen shortly. The configuration is compatible with other network monitoring tools, such as MRTG. By storing data for two years, one can easily figure out any long-term trends that appear.

The second component of the RADSL monitoring application is a graphical user interface. The interface is web-based, largely because it makes that data easily accessible to all users of the RADSL lines. This web interface is probably the component of the system that has undergone the most change, going from a simple, static web page to a fully fledged web application with dynamically generated content.

Initially the interface was fairly straight forward, based on ideas from a previous Masters student [Irwin, 2001]. This interface provided graphs for the up and downstream line speeds as well as a measure of the traffic on each of twenty RADSL lines. These twenty lines formed the initial infrastructure, with a further twenty four lines being added subsequently. The graphs were generated at fixed time intervals and were displayed on a static web page.

Since then, the interface has been significantly re-worked. It now monitors all forty-four of the Department's RADSL lines, with the facility to easily add more. Line speed, traffic, errors and discards are all monitored and graphed. All web pages are dynamically generated, and contain information extracted from the access multiplexers as the page is generated. An example of a web page from this system is given in Figure 3-3.

Figure 3-3. Sample page from RADSL monitoring application



As can be seen from Figure 3-3, this application displays a number of graphs. For each of the metrics (line speed, traffic, errors, and discards), four graphs are displayed. Each of these graphs displays the up and downstream values for the metric plotted against time. Each graph displays a different time frame in the same amount of space, being a day, a week, a month, and a year.

The information collected by this system is used for a number of purposes. It is used to determine which lines are most utilised, and this data is used to help balance the load between the blades on the access multiplexer. Each access multiplexer has a number of blades that are connected to external lines. Every line on a blade shares a common uplink from the blade to the rest of the network, so to get the best performance it is important to evenly distribute the load amongst the available blades.

When a line is faulty, the information collected from the system is used to determine when the fault occurred and the duration of the fault condition. If the line fault is reported to the local telephone

company (telco), this information is provided with the report to aid them in their diagnosis. Since this information can be compared against the dates on job cards in the area, *et cetera*, the availability of this data has often allowed the telco to quickly identify and rectify faults that would otherwise have taken a significant amount of time to trace.

The configuration of this application was done using the XML approach described in Section 3.2. It uses this configuration file to determine the host name to connect to, the SNMP community to use and the OIDs to retrieve from each hosts. A sample of this configuration is given in Figure 3-4.

Figure 3-4. Subset of configuration file

```
<?xml version="1.0" standalone="yes" ?>
<monitor>
  <host name="dslam1.ict.ru.ac.za" community="public">
    <oids>
      <oid name="upspeed" type="gauge" precision="32">.1.3.6.1.4.1.1795.2.24.2.6.8.1.1.1.1.6.$port.31</oid>
      <oid name="downspeed" type="gauge" precision="32">.1.3.6.1.2.1.2.2.1.5.$port</oid>
    </oids>
  </host>
  <host name="dslam1.ict.ru.ac.za" community="public">
    <oids>
      <oid name="communities">.1.3.6.1.2.1.47.1.2.1.1.4</oid>
      <oid name="interfaces">.1.3.6.1.2.1.2.2.1.3"</oid>
    </oids>
  </host>
  <host name="dslam1.ict.ru.ac.za" community="public@s1p1">
    <oids>
      <oid name="upspeed" type="gauge" precision="32">.1.3.6.1.4.1.1795.2.24.2.6.8.1.1.1.1.6.$port.31</oid>
      <oid name="downspeed" type="gauge" precision="32">.1.3.6.1.2.1.2.2.1.5.$port</oid>
    </oids>
  </host>
</monitor>
```

This format of configuration files makes it easy to extend the system to add new access multiplexers. The system itself is designed to automatically detect the presence of remote RADSL modems and will start logging data as soon as a new remote network is detected. This makes the system almost completely self-configuring once the initial information has been provided.

3.4. Summary

This chapter examined some of the typical problems associated with network monitoring. The first of these was the need to present information in such a way that it is useful to those people at whom it is aimed. Simplicity is often desirable, since the simpler the interface, the broader the spectrum of people who will be able to comprehend it.

The problem of multi-vendor networks that was introduced in Section 2.3.2 was revisited, and a solution was proposed. This solution uses an XML-based markup language to abstract from the vendor-specific

SNMP MIBs to an application-specific configuration file. This markup language was successfully used in two applications, which are described in Section 3.3 and Chapter 4.

A performance management application, the monitoring of RADSL remote access lines, was presented as an example of the problems associated with the SNMP MIB-II. RADSL is asymmetric in nature, making it an ideal case to examine the effectiveness of the XML approach presented in Section 3.2.

Chapter 4. Determining Network Topology

One of the problems that commonly occurs when networks are allowed to develop in an *ad-hoc* manner is that no network topology maps are produced, or those that exist are not updated. The lack of accurate topology maps makes pinpointing network faults more difficult, since it is difficult to understand which parts of the network relate to which other parts.

Many network monitoring applications will attempt to automatically determine the layout of one's network. They often achieve this quite successfully in a homogeneous network environment, but fail to do so in a hetrogeneous environment. This is because these products often use proprietary protocols such as the Cisco Discovery Protocol that was discussed in Section 2.2.5.

What is needed is a method to discover the topology of the local area network, irrespective of the layout or vendor of the various network devices. This is in essence what the various Internet mapping projects have attempted to achieve, only on a much larger scale.

4.1. Traceroute approach

With this in mind, the approach taken by one of these projects was investigated. This project used traceroute(8)-style probes to determine the path between any two hosts on the Internet [Cheswick, 1999]. This works much like the medical idea of *tomography*, that is the system attempts to build up a picture of the network by looking in from the edge of the network. This idea works very well on the large scale that the Internet mapping projects are examining, but, as far as the author can tell, had never been applied to networks of the local area network scale.

This approach looked very promising and an application was written that would record the network path to each of the 65536 hosts in Rhodes' class B network. The network path consists of a number of "hops", or devices that traffic must pass through in order to reach its destination. In any path, the last hop is the destination itself. Since this application was trying to extract routing information (the hops between the source and destination of a packet) rather than the location of each host on the network, the last hop of every trace was discarded. This left us with a record of only the hops between each host and the machine on which the application was running.

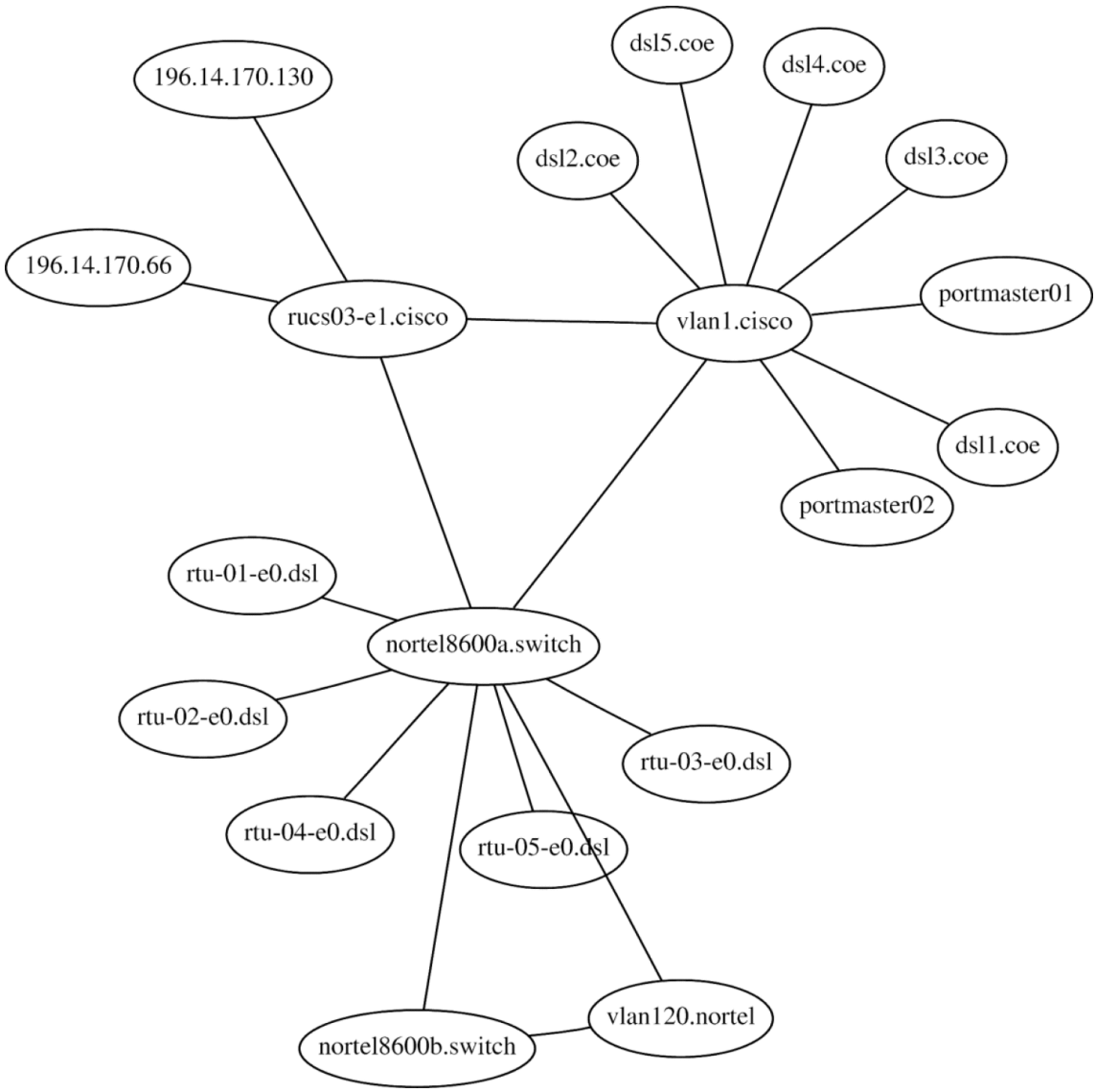
The application was run mid-week during the middle of the morning in the hope of capturing data on the majority of the computers on Rhodes' network (at this stage, there were no statistics on usage patterns). The information that was obtained from a run of the application was normalised to remove duplication before being written as a data file for a graphing utility.

The graphing application chosen to visualise the results of this test was AT&T Research Labs' GraphViz (`http://www.graphviz.org/`). This program is a sophisticated tool for laying out and drawing both

directed and undirected graphs. Its greatest feature is its ability to make very readable graphs from fairly complex data sets [Fowler, 2000]. Graphs are described in a language called “dot” and are then rendered by the application into a number of different vector and bitmap image formats.

Figure 4-1 shows an example graph created by this system. This graph shows the layout of Rhodes’ network at layer three of the OSI reference model.

Figure 4-1. Topological map of Rhodes University’s network



Some assumptions were made about the way networks work in order to draw a graph of the results. If a trace to a host did not contain at least one hop, the destination machine is on the same subnet as the machine running the trace application, and it is assumed to have the same default gateway as the machine running the application. While it is possible for two machines on the same subnet to use different gateways, this is not common practice.

It was also assumed that the forward and return routes of all packets are the same. This assumption allows one to derive the route between any two hosts based on the routes from a third machine to each of the machines in question by discarding all parts of the route that are common to both machines. While the assumption is generally true on a local area network, it is certainly not so on the Internet. However, without this assumption, it would not be possible to map the layout of a network in this manner. For this reason, this assumption is used by all Internet mapping projects [Cheswick, 1999].

The results of this attempt are shown in Figure 4-1. There are two interesting features about Rhodes' network that this graph highlights: Firstly, the graph is a lot smaller and flatter than was originally expected and secondly, there are two loops formed around nortel8600a.switch.

Rhodes' network is significantly more complicated than Figure 4-1 shows. This complexity, however is at layer two (datalink) of the Open Systems Interconnect (OSI) 7-layer model [Briscoe, 2000]. Tracing a route to devices in the way this application has done only detects devices that affect the TCP time to live (TTL) field — that is, devices at layer three (network) of the OSI network model. There are very few layer three devices at Rhodes, which explains the relative simplicity of the graph that was produced.

The two loops are more interesting. The loop between nortel8600a.switch, nortel8600b.switch, and vlan120.nortel occurs because of the assumption that hosts with no intermediate hops were on the same subnet. This assumption is true except in the case where the IP address being traced to is a different interface on the default gateway itself. Figure 4-2 illustrates this for the case where a machine's default gateway is vlan120.nortel (shown in line 2 for a machine called omniscient). To be the default gateway, vlan120.nortel must necessarily be on the same subnet as omniscient. This means there will only be one hop from omniscient to vlan120.nortel, as is shown in lines 3–5.

While both omniscient and vlan120.nortel are on the 146.231.120.0/21 subnet, nortel8600b.switch is not — it is on the the 146.231.128.0/21 subnet. There is, however, only one hop between omniscient and nortel8600b.switch (shown in lines 6–8). This violates the afore-mentioned assumption, and occurs because nortel8600b.switch is an interface on the same layer three switch as vlan120.nortel. In this case, the switch transparently routes packets between its interfaces, causing the apparent loop.

Figure 4-2. Trace showing the default gateway assumption

```
1  guy@omniscient:~% netstat -r | grep default
2  default            vlan120.nortel    UGSc       12 100257463   fxp0
3  guy@omniscient:~% traceroute vlan120.nortel
4  traceroute to vlan120.nortel (146.231.120.1), 64 hops max, 40 byte packets
5   1  vlan120.nortel.ru.ac.za (146.231.120.1)  0.983 ms  0.969 ms  0.874 ms
6  guy@omniscient:~% traceroute nortel8600b.switch
7  traceroute to nortel8600b.switch (146.231.128.210), 64 hops max, 40 byte packets
8   1  nortel8600b.switch.ru.ac.za (146.231.128.210)  1.080 ms  0.926 ms  0.877 ms
9  guy@omniscient:~%
```

The second loop between nortel8600a.switch, vlan1.cisco, and rucs03-e1.cisco is caused by the assumption that the forward and return routes are the same. Figure 4-3 shows how this assumption breaks down.

Lines 2–5 of this figure show that vlan1.cisco knows that it should use its default route (0.0.0.0/0) to forward all traffic destined for the Internet to rucs03-e1.cisco (146.231.128.206). In the same way, nortel8600a.switch also forwards all outgoing traffic bound for the Internet to rucs03-e1.cisco, as is shown by line 13.

rucs03-01.cisco, on the other hand, forwards all incoming traffic from the Internet to nortel8600a.switch (146.231.128.205) irrespective of its final destination within the University's network (146.231.0.0/16). This is shown by lines 16–19 of Figure 4-3. In this case, nortel8600a.switch is tasked with the correct forwarding of that traffic to other devices, including vlan1.cisco.

Figure 4-3. Routing table entries showing the return route assumption

```
##  vlan1.cisco.ru.ac.za has address 146.231.135.22
1   vlan1.cisco>show ip route 0.0.0.0
2   Routing entry for 0.0.0.0/0, supernet
3     Known via "static", distance 1, metric 0, candidate default path
4     Routing Descriptor Blocks:
5     * 146.231.128.206
6         Route metric is 0, traffic share count is 1

##  nortel8600a.switch.ru.ac.za has address 146.231.128.205
7   NORTEL8600A:5# show ip route info 0.0.0.0
8   ================================================================
9                             Ip Route
10  ================================================================
11      DST     MASK              NEXT COST VLAN  PORT  PROTO  AGE
12  ----------------------------------------------------------------
13  0.0.0.0  0.0.0.0  146.231.128.206     1    1   -/- STATIC    0
14  1 out of 28 Total Num of routes displayed.

##  rucs03-e1.cisco.ru.ac.za has address 146.231.128.206
15  rucs03-e1.cisco>show ip route 146.231.0.0
16    Routing entry for 146.231.0.0/16
17    Known via "static", distance 1, metric 0
18    Routing Descriptor Blocks:
19    * 146.231.128.205
20       Route metric is 0, traffic share count is 1
```

In other words, traffic from a device connected to vlan1.cisco to the Internet takes the most direct route it can, whereas traffic from the Internet to the same device passes through an extra hop, namely nortel8600a.switch. This means that the loop shown on the Figure 4-1 does in fact exist, but in practice it is uni-directional rather than bi-directional as indicated on the graph. The routing configuration that caused this anomaly was a temporary optimisation while systems were being migrated from vlan1.cisco to nortel8600a.switch. vlan1.cisco has subsequently been decommissioned and this loop has been removed.

4.2. SNMP approach

The previous section looked at the topology of a network at layer three of the OSI network model. As has been shown, detecting layer three devices is fairly straightforward since these devices alter the packets that are transmitted on the network. Layer two devices, however, do not make any changes to packets on the network. As a result, these devices are essentially transparent, making them impossible to detect using the techniques described in Section 4.1.

To understand how to go about detecting layer two devices on a network, one first needs to understand something about these devices. Devices that operate at the datalink layer typically come in two forms: managed and unmanaged. Unmanaged devices simply plug into the network and operate as is — they are not configurable in any way.

Managed devices, on the other hand, are configurable in some way. Typically, these devices bind their own IP address and are network addressable, meaning that they can be remotely configured. Vendors are free to choose the means and protocols that are used to configure these devices, but for reasons of inter-operability, these devices usually contain an embedded SNMP agent. These SNMP agents can be addressed using the MIB-II, meaning that a large proportion of the information available from them is available in a standardised form.

Two of the metrics that are standardised in MIB-II are the interface and IP tables. These tables contain, amongst other things, a list documenting which MAC addresses (`interfaces.ifTable.ifEntry.ifPhysAddress`) and IP addresses (`ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaIfIndex`) are associated with each interface on the device. The information contained in these two tables can be used to determine where a particular device is located on the network. By making some assumptions about the entries in these tables, a layer two topology map can be built.

On any layer two device, one has a number of interfaces. On each of these interfaces, one can have a host or another network device, in other words, a leaf or a node on a graph. Since these devices are transparent on the network, the difficulty lies in deciding whether each interface is connected to a leaf or to another node. Some progress can be made by assuming that any interface that has only one MAC address or IP address associated with it, is a leaf of the graph. This is a reasonable assumption since computers typically only have one network card and one IP address.

The problem lies in deciding what to do with interfaces that have multiple entries. For example, computers that function as web servers often have multiple IP addresses associated with them. Fortunately, it is reasonably rare for these computers to use more than one network card on a particular network, meaning that they only have one MAC address associated with them. In the same way, computers that use IEEE 802.3ad link aggregation to connect several network cards to a single network typically use one MAC address shared between all aggregated network cards. From this, one can assume that any interface that has both multiple MAC addresses and multiple IP addresses is a node, and any interface that has just one MAC address or one IP address associated with it is a leaf.

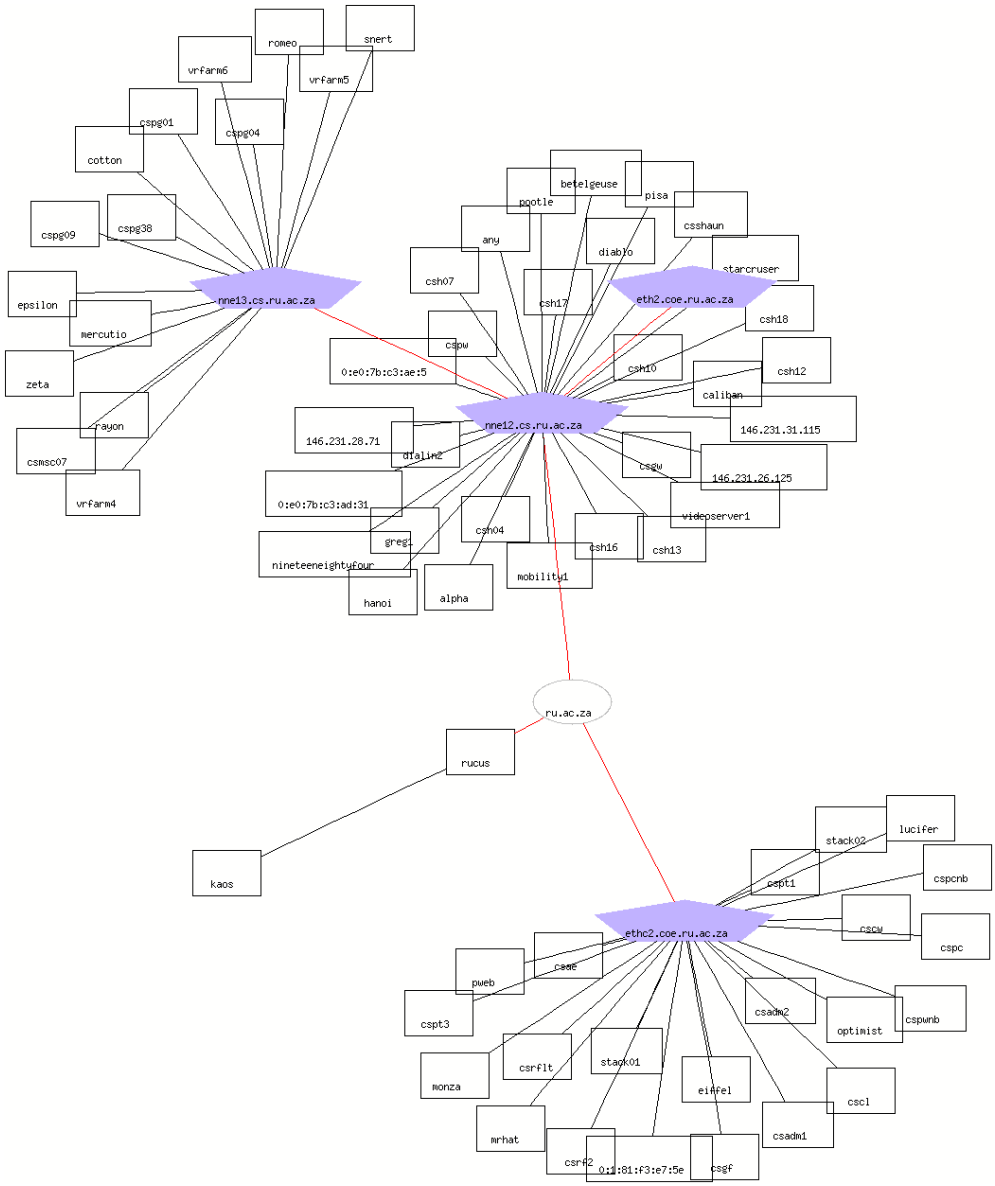
Once there is a way of separating nodes and leaves on the graph there is the further problem of determining the IP address of the managed layer two device that forms the node. There are two approaches that can be used to solve this: either some seed information can be provided about which devices on a specific network are network devices or the graphing application can attempt to make an intelligent decision about what devices are network devices.

While the first of these approaches is by far the simplest, the latter is not as complex as it might at first seem. The Institute of Electrical and Electronics Engineers (IEEE) maintains a list of Organizationally Unique Identifiers (OUIs). These OUIs form the first twenty-four bits of every MAC address, and are used to identify the manufacturer of the device [IEEE, 2002]. To a large extent, vendors that make network devices such as managed switches do not make network cards, and vice versa. Those vendors who manufacture both types of devices tend to have multiple OUIs (Cisco Systems, for example, have registered 192 OUIs), and use a different OUI for each class of device.

If the graphing application is provided with a list of OUIs that are known to belong to network devices, it can use this information to intelligently decide which of the devices on a network should be considered to be layer two devices, and thus represented as nodes on the graph. It can further use this information to work out the IP address of each node, which it can use to recursively query each node for its interface table.

Based on these assumptions, a layer two network mapping application was written. This application again used the GraphViz graphing engine. This application was used to plot a topology map of the Computer Science Department's network (this smaller network was chosen because it had a known topology, allowing the accuracy of the results to be determined). Seed information about the layout of the network was provided using an XML-based configuration file as described in Section 3.2. The results of a run of this application is shown in Figure 4-4.

Figure 4-4. Layer two topology map of the Computer Science Department



In Figure 4-4, the pentagons represent layer two network devices, the squares represent computers on the network, the oval in the center represents the up-link to the rest of Rhodes' network and red lines represent up-links between switches.

There are two abnormalities in this graph that are worth noting; eth2.coe.ru.ac.za is a layer two switch that is shown on the map to have no hosts attached for it. This apparent lack of leaves occurred because at the time this map was generated, the SNMP agent on that node was not running. The result is that all the leaves that should be rendered as attaching to eth2.coe.ru.ac.za are shown as attached to nne12.cs.ru.ac.za.

rucus is shown on the map to be a computer, but has another computer, kaos attached to it. This layout is correct; rucus has two network cards in it and acts as a bridge (using FreeBSD's bridge(4)) between

kaos and the rest of the network.

The biggest problem with the approach taken in this section is that it does not scale well. While it was known that the approach would never work on the scale of the Internet, it was assumed that the method could be used to represent a reasonable sized campus network. An attempt was made to graph the whole of Rhodes' campus using the SNMP method outlined above. The graph produced had an order of magnitude more hosts on in, and took just under two hours to lay out and render. Unfortunately the rendered graph was completely illegible, which made it useless as a representation of Rhodes network. Clearly another way needed to be found to represent information about a network of this magnitude. Chapter 5 looks at another, more successful approach.

4.3. Summary

This chapter examined two approaches for determining the topology of a network. The first of these, using traceroute(8) detects network devices that operate at layer three of the OSI network stack, and the second, SNMP, detects devices at layer two. Both these applications were built on the XML framework described in Section 3.2. These applications partially fulfil the performance and configuration management requirements of the OSI network management model.

The information obtained using these applications can be used to find the location of hosts on a network, as well as to predict the growth rates of a network. While the traceroute(8) approach works at a very large scale (as demonstrated by the Internet mapping projects), the SNMP approach does not scale well. For this reason, alternative approaches were investigated. Chapter 5 will look specifically at the problem of predicting growth rates, while Chapter 6 will re-examine the challenge of locating hosts on a network.

Chapter 5. Tracking Network Growth

One of the uses mentioned for the application in Chapter 4 was to predict network growth. Unfortunately, the approach taken by the previous chapter was not really sufficient to provide the data required to make accurate predictions of network usage and growth. Part of the reason for this is that it does not record historical data to allow for comparisons over time.

In order to get an accurate picture of network growth and utilisation, a new approach was required. One of the key features of such an approach would be that it should keep historical information on the utilisation of various parts of the network.

5.1. Design issues

At a high level, the approach taken by this application is straightforward. Every half an hour, it collects information on which hosts are available on the network and records that information to a database. Logistically, however, things are more complicated than that.

Rhodes has a class B network block assigned to it by ARIN, the American Registry for Internet Numbers. In class B networks, the first two octets of an IP address are fixed and the last two are variable. In other words, the first 16 bits of the 32 bit IP address space define the owner of the network block, and the last 16 bits define hosts on the network. This gives 65536 available IP addresses [BCP 12]. In Rhodes' class B, the first two octets correspond to the dotted-decimal representation 146.231, meaning that anything matching 146.231.*.* is considered to be on Rhodes' network.

It was estimated from entries in the DNS tables from Rhodes that about 4500 IP addresses were in use at Rhodes, or roughly seven percent of the available IP space. (It will be shown later in this chapter than this estimate was a little generous.) Rhodes' network is subnetted according to Tsuchiya's scheme for assigning subnet numbers [RFC 1219], which means those IP addresses that are in use are scattered throughout the class B network rather than being a contiguous block within the available IP addresses. This makes finding an IP address that is actively in use akin to finding the proverbial *needle in a haystack*.

The simple solution is to ping every host on the network block and see which ones respond [ping(8)].

If, in order to do a scan of the network, a single 64 byte ICMP ping packet is sent to each of the 65536 hosts, the result is four megabytes [1] of outgoing network traffic. In addition, from the estimate above, the application can expect 4500 replies to these ICMP echo requests, adding another 280 KB to the amount of data on the network. Together, this represents a significant amount of data generated by this method, and the sheer volume of traffic led to some implementation problems, as shall be seen in Section 5.3.

1. (64 * 65536) B / 1024 KB / 1024 MB = 4.0 MB

For such a probe to gather useful information, it has to be repeated at regular intervals. Since this is an active probe (as explained in Section 2.2.3), there is a trade off between usefulness and invasiveness. If the experiment is run too often, it risks saturating the low-bandwidth shared-media segments of the network. This is something that needs to be avoided, and something that was considered during the implementation of the probe.

5.2. Implementation

The first attempt at an implementation was a single Perl programme that used Perl's Net::Ping module. The program attempted to sequentially ping every host in Rhodes' class B network once every half hour. The IP addresses of those hosts that were visible on the network were written to a text file, which served as a database of hosts that had been seen. The data obtained from this probe was also fed into a round-robin database of the type mentioned in Section 3.3.

This approach suffered from a serious limitation; it took longer than half an hour for each run to complete, leading a situation where the monitoring machine was constantly sending out ICMP packets, and where there was overlap between two consecutive runs. The presence of overlap caused the most problems, leading to, on occasion, corruption of the data file when two processes attempted to write to it simultaneously. Clearly a better solution was required.

Realising that the excessively long time this program was taking to run was caused by the process blocking while waiting for an ICMP echo reply, rather than network saturation, led to the first improvement. By splitting up Rhodes' class B into smaller blocks, these blocks can each be passed to a separate worker process, allowing multiple IP addresses to be queried simultaneously. This is a weak attempt at parallel processing, and brings with it the associated Inter-Process Communication (IPC) difficulties.

Attempts were made with various file locking and IPC methods (such as shared memory) to ensure that the data from each of the worker processes was correctly stored, and that data was not lost or corrupted. The KISS principle (*Keep It Simple, Stupid*) prevailed, however, and the problem of IPC and locking was offloaded to a SQL database package, namely MySQL.

MySQL was chosen because it is generally accepted to be a very fast database for cases where data integrity is not of utmost importance [MySQL, 2001]. In the case of this application, speed was certainly of more importance than integrity, since there were likely to be a large number of database INSERTs in a very short period of time. The reason that integrity is less important than speed is that the network itself is a lossy medium, meaning packets get lost in the normal course of operation. So long as the database performs each INSERT atomically (which all database management systems guarantee) and each record is limited to a single INSERT it does not particularly matter if a relatively small number of these INSERTs fail. In practice, however, this never happened. Another reason for preferring MySQL over other database products is its ability to handle large numbers of simultaneous connections well, which is an important consideration for an application that is going to create a large number of child

processes.

Several experiments were done to determine what the optimal number of child processes would be; too many of them and the machine's processor would be overloaded, and too few of them would increase the time taken for a run to be processed. Since the machine running this application was used for other development work, it was important not to compromise the usability of the machine, making a longer run more preferable. In the end, Rhodes' class B network was divided into 128 blocks, with each block being processed by a separate worker process. These blocks were created as 23 bit networks on CIDR boundaries allowing them to easily be manipulated by the program [RFC 1519]. [2] This configuration seemed to give the best time/load performance.

The round robin database was maintained even after the switch to a SQL database backend. It was found that real-time generation of graphs for the web interface described in Section 5.3 performed better when linked to an RRD database than it did when it was connected directly to the MySQL database. To facilitate this, another program was developed to extract records from the MySQL database and insert them into the RRD. This was done every half an hour, and is offset from the data gathering program by fifteen minutes.

Once the initial configuration and performance tuning had been done, this application was left to run on Rhodes' network for just over a year. The results that were gathered from this year-long run are discussed later in this chapter.

5.3. Problems and solutions

Any application of this scale is bound to run into some problems along the way, and this network monitoring tool was no exception. These ranged from social issues to networking problems, and this section aims to document some of the more interesting ones.

5.3.1. Social considerations

One of the first problems encountered was social rather than technical. There are a few people on campus who run personal firewall software on their machines, and some of these people have their software configured to report on all unknown traffic. One of these people noticed ICMP echo requests coming from the monitoring machine and decided to investigate the matter. In a Usenet post to the University's ru.chat newsgroup, he asked if anyone knew the reason for the probe.

Several posts followed in which it was explained that the probes were harmless, useful, and formed an insignificant amount of traffic to his machine (128 bytes an hour). One of the comments that came out

2. A CIDR boundary is a place where, when represented in binary, all the bits to the left of the boundary do not vary, whilst those to the right of the boundary do. This gives a range of addresses in a network. It is efficient to implement since a simple AND can be used to determine whether a host is on the network.

of the discussion was:

> When a stream of pings arrive at my UTP socket it would be nice to know why *before* it happens, not after, especially in the light of nimba (sic) and friends.

In the light of this and in consultation with the Systems Manager at Rhodes, it was decided to change the source address of all ICMP echo requests to more accurately reflect what the purpose of them was. This new source address had a reverse DNS entry of `we.are.drawing.network.maps.of.ru.ac.za` to ensure that it was obvious that the probes were intentional. In addition, a web page was set up on the `we.are.drawing.network.maps.of.ru.ac.za` domain informing people of exactly what the *modus operandi* of the project was.

Since this first query about the project, it has run for over a year with no further comment. It can only be assumed that those people who have noticed the requests have been satisfied with the explanation available on the web page.

5.3.2. Network cards

Early on in the experiment, the monitoring machine experienced problems in transferring the ICMP echo requests it generated on to the physical network. This was characterised by errors in the system log indicating a lack of memory buffers (`mbufs`). Some research indicated that the likely reason for these errors was the SMC 1211-TX network card that was in the machine. The SMC 1211-TX is based around the RealTek 8139 chipset, which is known to experience problems handling a high load.

The problems with the RealTek network interface controller (NIC) chipset are best described by Bill Paul's commentary in the source code for FreeBSD's rl(4) driver. The first two paragraphs of this commentary are reproduced below [Paul, 1998].

> The RealTek 8139 PCI NIC redefines the meaning of 'low end.' This is probably the worst PCI ethernet controller ever made, with the possible exception of the FEAST chip made by SMC. The 8139 supports bus-master DMA, but it has a terrible interface that nullifies any performance gains that bus-master DMA usually offers.
>
> For transmission, the chip offers a series of four TX descriptor registers. Each transmit frame must be in a contiguous buffer, aligned on a longword (32-bit) boundary. This means we almost always have to do mbuf copies in order to transmit a frame, except in the unlikely case where a) the packet fits into a single mbuf, and b) the packet

> is 32-bit aligned within the mbuf's data area. The presence of only four descriptor registers means that we can never have more than four packets queued for transmission at any one time.

The second paragraph is perhaps the most significant. It comments that one cannot have more than four packets queued for transmission. For an application that is trying to generate 65536 ICMP echo requests in batches of 128 at a time, this limitation is likely to have a noticeable effect.

With this in mind, the network card in the monitoring machine was replaced with an Intel Pro/100S card (sometimes also known as an EtherExpress Pro 10/100S, but this use is deprecated by Intel). This card was chosen because it is known to work very well with FreeBSD — the operating system on which this application was being run — and is extensively used on the FreeBSD Project's own network [FreeBSD, 2002].

The result was an immediately noticeable increase in performance and throughput. The error messages regarding the lack of `mbufs` disappeared, leading to the conclusion that the problem was indeed related to the use of a RealTek based network card.

5.3.3. MySQL problems

As was mentioned in Section 5.1, MySQL was chosen to be the database engine for this application because its performance was known to be substantially better than other open-source databases. The MySQL backend performed very well for the first five months of the project, easily handling the seventy thousand or so records that were inserted each day.

Unfortunately, as the database grew larger, performance slowly degraded. When the record count reached around eleven million, the performance of the database had become so bad that it was taking longer to update the indices on the database than it was to generate the data in the first place. This loss of performance was unacceptable because it affected the usability of the machine that was doing the monitoring.

An investigation with FreeBSD's truss(1) application determined that the performance loss in the database engine was due to the system blocking on disk writes. The machine running the monitoring application has a single 20 GB Western Digital IDE hard drive in it, running at 5600 RPM. This drive is slow by modern standards, so it was conceivable that using a different hard disk would fix the problem.

Lacking a suitable replacement, another solution had to be found. As an interim measure, the existing eleven million records in the database were moved to offline storage in early April 2002. The contents of the database were then deleted, restoring the performance of the database engine and the machine as

a whole. Since the contents of this database had already been summarised in a round robin database, this rather drastic solution had very little noticeable impact on the outputs of the application.

As the next six months passed, the database again grew to be over ten million records. The resultant loss in performance was repeated, to the point where another solution needed to be found.

This time, the records were moved to a new machine. This machine had a series of SCSI hard disks in a RAID 5 configuration, offering significantly better disk performance than the original machine.

The original eleven million records were also imported into this new database, resulting in a single MySQL database that had a little over twenty-one million records in it.

In order to test the performance of this new database, some complex queries were run against it. These same queries were also run against the existing ten million records on the original machine in an effort to provide a comparison. The difference between the two results was significant. A query that took a little over an hour on the original machine took just over fifteen minutes on the machine with SCSI drives. This clearly demonstrated the need for high performance disk drives in machines running heavily utilised databases.

5.3.4. Routed subnets

One of the assumptions made while determining the impact of this monitoring application on the network's bandwidth was that all backbone links had sufficient bandwidth to carry the ICMP requests for all the machines they were serving. This assumption is reasonable since, as was explained in Section 2.4.1, the majority of the inter-switch links at Rhodes run at between 100 Mbps and 1 Gbps.

On two occasions during the year, this assumption was invalidated. Both of these occasions involved large subnets being routed over a dial-in line.

The first was a conference which Rhodes hosted at a local hotel. Connectivity for the conference was provided by a 64 Kbps ISDN dial-in line. This line had a class C network (256 IP addresses) routed over it, of which approximately ten were in use. Fortunately the problem was pre-empted and the ISDN router that provided the connection onto Rhodes' local area network was configured to reject all ICMP requests from the monitoring machine. This effectively prevented the ICMP queries from interfering with the remote dial-in network.

Unfortunately this was not so in the second case. This time an entire twenty-one bit network was routed over a 33.6 Kbps analogue dial-in connection between Grahamstown and Johannesburg. Of these 2048 IP addresses, under ten were actually in use, so in many ways, this routing was overkill.

Once every half hour, the monitoring system attempted to send 128 KB of data down this line. The result of this traffic is that for thirty seconds during the run, the data line is completely saturated by the ICMP echo requests. This delay was noticeable to users of the dial-in line who were attempting to communicate with Rhodes using interactive protocols such as telnet or ssh.

After a brief telephone call with the Systems Manager, the problem was resolved by configuring the monitoring system to simply ignore the subnet in question for the duration of the event.

5.4. A web front-end

During the course of the experiment, requests were made that the data gathered was made available to people. To facilitate this, a web-based interface to the round robin database was developed. This front-end allowed the user to generate a number of graphs to compare the number of hosts on any given subnet over a particular time interval.

Like the graphs of Section 3.3, the graphs in the application were generated using Tobias Oetiker's rrdgraph(1) graphing engine. The graphs produced by this application, however, are significantly more complex to generate than those of Chapter 3 since they allow the user to choose the time interval over which they are generated, and to overlay information about various subnets onto the same graph. The advantage of this system is that it easily allows the user to compare different subnets.

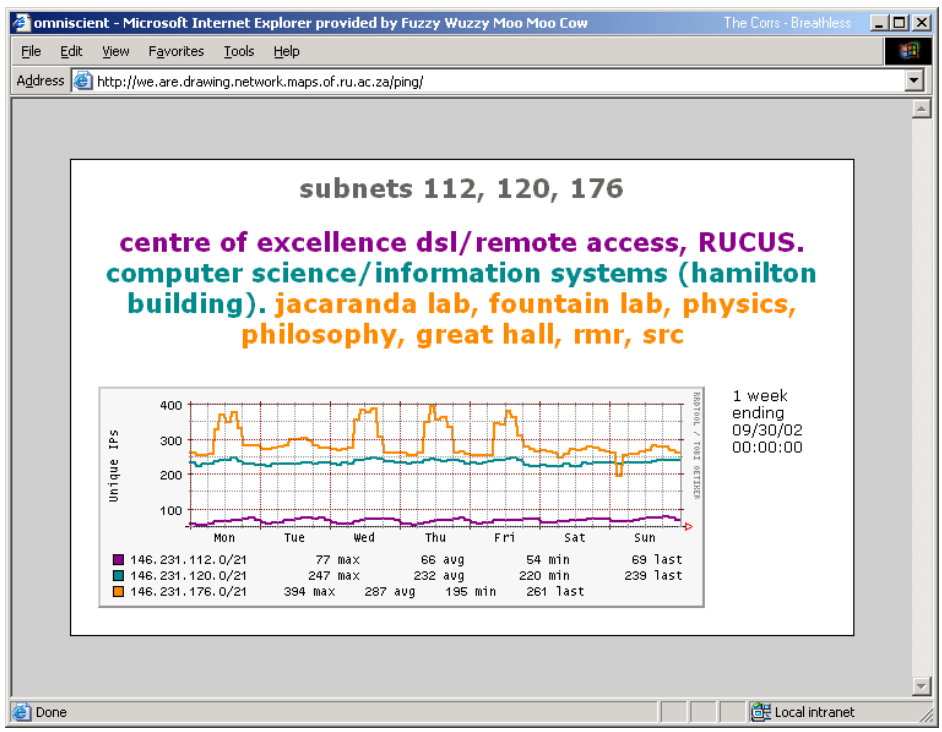
Figure 5-1 shows how the application can be used to provide a comparison between three different subnets — in this instance, the `146.231.112.0/21`, `146.231.120.0/21` and `146.231.176.0/21` subnets. It also illustrates the method used to determine the colour [3] of each line on the graph.

Using this web-based application, a number of interesting patterns in the usage of computers at Rhodes was discovered. Figure 5-1 shows one such pattern. Unlike the other two subnets shown, usage of `146.231.176.0/21` subnet clearly shows a tendency by some people to only have computers switched on during working hours. The computers on this subnet that are shown to be always switched on are probably those computers in the public computer laboratories, which are available to students twenty-four hours a day. The low peak in the graph on Tuesday 24 September can be simply explained and is interesting in itself; it was Heritage Day (2002), a South African public holiday.

3. One of the more interesting problems encountered while trying to design a system that generated comparative graphs for various subnets was that of choosing colours for the various lines on the graph. In theory, there could be up to thirty-two lines on each graph. In order to display each of these lines clearly, the colours assigned to each of the lines need to contrast as much as possible. The algorithm used to select these contrasting colours is described below:
Using an additive colour wheel, thirty-two colours were selected from even spaced intervals around the wheel. These colours included the three primary colours (red, green and blue — RGB), as well as a range of intermediate colours between each of the primaries. Each of the colours chosen had approximately the same level of brightness. These colours were transcribed into an array using their hexadecimal RGB notation. The arrangement within this array mimicked each colour's relative positions to each other on the additive colour wheel. The start and end position were essentially arbitrary but, for aesthetic reasons, they were chosen so that blue was in the middle of the array.
When a number *n* of colours had to be chosen for use on a particular graph, a step value *s* was calculated using the formula *s = 32 / (n * 2)*. Starting at the colour whose array index was *s*, *n* colours were chosen from the array in such a way that their array index was *s* greater than that of the previous colour.
This resulted in a set of opposing colours that are evenly spaced around the additive colour wheel, giving us a range of easily distinguishable colours for the graph.

This trend towards only having computers on during office hours can be seen in other subnets too. An interesting observation is that it tends to be more pronounced in those subnets serving departments and divisions that are less computer-centric. At the extreme ends of this observation are the subnets serving the Computer Science department, and the subnet serving the University's Sports Administration, Estates division, and Grounds and Gardens division. The Computer Science department tends to have more computers on the network outside of office hours than within, while the Estates division has almost no computers on outside of normal office hours.

Figure 5-1. Comparison of subnets



5.5. Some results emerge

As was mentioned at the beginning of this chapter, the chief aim of this application was to enable Rhodes to determine the growth rate of computers on various parts of its network. This information is important for a number of reasons: It allows the University administration to predict the number of computers that will join the network in a given time period and to budget for resources, such as bandwidth, accordingly. It also provides an indication of which areas of the network are most used, and allows for planning of strategic upgrades of the backbone infrastructure to handle this additional demand.

Information on the number of hosts on various subnets at Rhodes was gathered for just over a year, and the results of this experiment are summarised for a sample of subnets on a monthly basis in Table 5-1.

This table shows the number of hosts on Rhodes' campus as a whole (all), as well as four of the thirty-two subnets on campus — namely the admin subnet (Grounds and Gardens, Estates division and Sports Administration), the hamilton subnet (housing the departments of Computer Science and Information Systems), the resnet subnet (including all student residence networking on campus), and the campus subnet (containing several of the public laboratories on campus). Three of these subnets have already been mentioned in Section 5.4, and will now be examined in more detail.

Table 5-1. Number of hosts on Rhodes University's network

month	all	admin	hamilton	resnet	campus
October 2001	2521	122	123	60	351
November 2001	2614	128	121	57	360
December 2001	2367	123	124	10	346
January 2002	2349	131	132	9	458
February 2002	2761	129	282	75	535
March 2002	2749	129	301	71	541
April 2002	2907	126	314	129	555
May 2002	3018	122	306	150	564
June 2002	2891	126	295	133	543
July 2002	2917	132	296	134	545
August 2002	2962	128	298	165	554
September 2002	3037	126	297	171	556
October 2002	3165	129	291	172	541
raw growth rate	26.224	0.280	7.810	6.266	3.630
corrected rate	22.560	0.130	0.292	5.890	1.582

A cursory glance at the figures presented in Table 5-1 shows a number of trends. On the campus as a whole, the number of IP addressable hosts has increased over the year. The drop in the number of hosts over June and July is explained by the University's mid-year vacation.

The resnet subnet shows a large growth in numbers from the beginning of the year. This is explained in part by the fact that students vacate their residence room over the end of year vacation, and in part by a significant increase in the number of network-connected residences on campus.

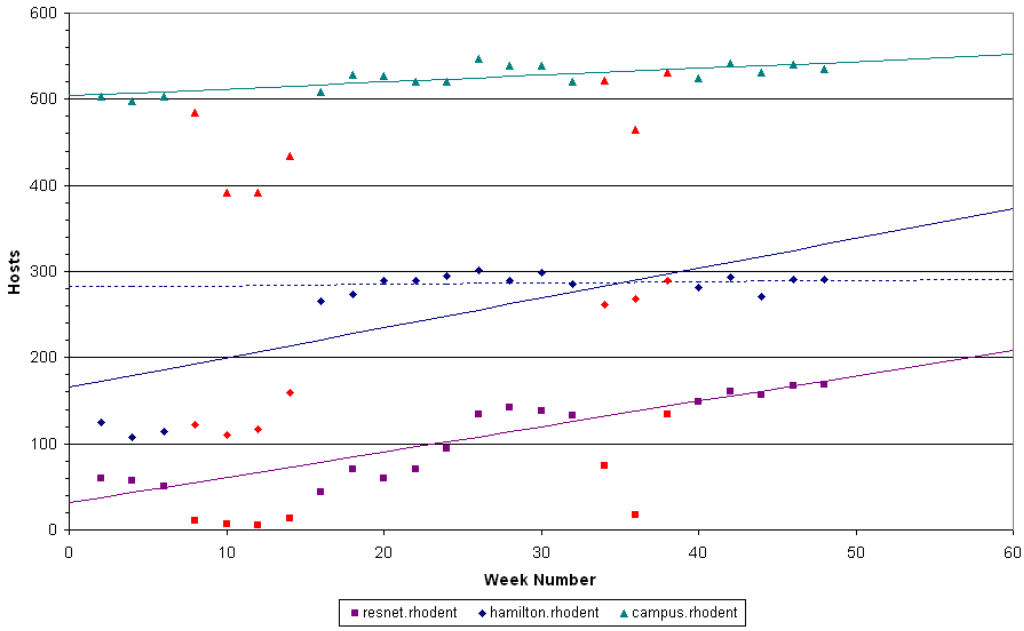
Both the campus subnet and the hamilton subnet show sudden growth spurts over a short period of time. This growth occurs as large laboratories of computers are brought online. In the case of the hamilton subnet, this was two new undergraduate laboratories in January. On the campus subnet a number of new laboratories have been commissioned over the course of the year.

One of the few subnets on campus that has shown very little increase in the number of network addressable hosts is the admin subnet. This subnet also seems immune to the inconsistancies introduced by the University's terms.

Analysis of this data more clearly shows this proclivity. A chart of the number of hosts versus time can be plotted, and the slope of this chart will give an indication of the growth rate. Normal regression techniques can be used to calculate the average slope of this chart, which provides a corresponding growth rate. Accurate regression requires more points than are presented in Table 5-1, so this data was recalculated on a bi-monthly basis.

Unfortunately the presence of University vacations in the data skews the resultant slope. The student population at Rhodes drops significantly over these periods, and this has a corresponding effect on the number of computers in use at the University. The resnet subnet, for example, is almost completely devoid of computers during the December vacation. These drops in numbers and the resultant extreme outliers are caused by a known phenomenon, and so they have been excluded from the chart shown in Figure 5-2

Figure 5-2. Growth rate of Rhodes University's network



The chart above shows the growth rates for three subnets on campus. The red points represent data collected during the University vacations, and were excluded from the calculation of the slope for each series. The campus subnet is typical of the growth rates seen at the University, whereas the resnet subnet shows a particularly high growth in this area. This relatively steep slope is backed by empirical evidence

and is likely to continue as more residences get connected to the University's network.

The hamilton subnet shows one of the inherent traps in analysing this data. Hamilton Building was only finished in late 2001, and only obtained full occupancy at the beginning of the year. This can be seen by the significant differences in the numbers of hosts between week seven and week fifteen. There are two slopes plotted for this series — the solid line represents the slope taking into account all the available data, and the dashed line represents the slope taking into account only data obtained during 2002.

The noticeable difference between these two slopes emphasises the necessity to take into account known events and abnormalities that occur during the course of monitoring. It is only by considering these events carefully that a true picture of the growth rate can be achieved. This sudden increase in numbers as the building gained occupants would become less noticeable as the period of monitoring increased. Unfortunately, the year's worth of data available for this analysis is too short a period to smooth out minor disruptions in the long-term trend.

5.6. Unexpected uses

The data gathered by this application proved to have two additional uses that were not expected. The first of these proved to have significant use on campus.

5.6.1. DNS dead wood

One of the problems associated with maintaining a large network is management of entries in the Domain Name System (DNS). As the network evolves, new machines are assigned DNS records and old machines cease to exist. A good DNS management policy needs to take this in to account.

While provision is made for the addition of records at Rhodes, historically entries have never been removed from the DNS database after they cease to be useful. One of the main reasons for this is that, while people are compelled to approach the DNS administrator to get a new registration, they often neglect to inform him when the registration is no longer requires.

Over the years a significant amount of *dead wood* has built up in Rhodes' DNS database to the point where there are now in excess of 4500 entries in the database, serving a little over 3000 hosts. This means approximately a third of the entries in Rhodes' DNS database are no longer used.

Since the application described in this chapter has been gathering information on what hosts are accessible on Rhodes' network, the data it has acquired can be used to prune some of the *dead wood* from the DNS database. Any host with a record in the database that has not responded to at least one ICMP echo request within the last thirteen months is almost certainly no longer in use, and any host that has not responded in the current year is also probably no longer valid.

A list of hosts together with the last date on which they were seen was extracted from the MySQL database used by this application. This list was compared with the list of hosts currently in the DNS database to produce a new list of hosts that had not been seen within the specified time period. This list was analysed to remove any obviously erroneous records (such as subnet network addresses), and was then given to the Systems staff to further analyse before they removed the old records from the DNS database.

This falls under configuration management of the OSI model.

5.6.2. Uptime indication

Many network administrators strive for the much touted “Five Nines”, or 99.999% availability [Pitt Turner, 2001]. This application provides a way to measure the network availability component since it keeps a record of every time a host responds to an ICMP echo request. In other words, it keeps a record of the availability of every host on the network. This availability can be calculated for any host in the database over a given time period.

Obviously an accurate availability measurement requires that the machine is constantly monitored, rather than tested every half an hour as this application does. Such a measurement also requires that the monitoring machine be available all the time. Both these requirements are unrealistic in terms of this application, but that does not prevent the application from providing an “educated guess” of the availability of a host.

There are two availability figures that can be calculated from the information in the MySQL database. The first is a theoretical availability — that is a calculation of the number of times a particular host should have theoretically been seen in any given time period compared to the actual count of the number of times it was seen in that period. This approach does not take into account any times the monitoring machine was unable to query the host, and so is the less accurate method.

A more accurate way of calculating this relies on the fact that, while it is querying the availability of hosts on the network, the monitoring application queries the availability of the machine on which it is running. Since packets destined for the local host are passed through the loop-back device rather than being passed over the network, it can assumed that this query is always successful. If the number of times the monitoring machine was visible is compared to the number of times the host in question was visible over any given time period, the result is a measurement of the actual availability of the host.

As a performance management application, these methods of determining the network availability of specific hosts on the network may provide systems administrators with useful information when tracing faults or determining which of their machines are the most reliable. The network availibility of hosts also provides an indicator of how well particular systems are adhering to service level agreements.

5.7. Summary

This chapter looked at an active monitoring application (in the performance management area of the OSI model) that tried to predict the growth rate of a network. While this method was more accurate than simple empirical analysis of the maps presented in Chapter 4, it is not without problems of its own. Some of these problems, such as those described in Section 5.3 were easily solved.

Analysis of the results in Section 5.5 showed that there were pitfalls inherent in the interpretation of the data that had been gathered. These problems are more difficult to overcome and require careful preparation and analysis of the data to ensure accurate results.

While this chapter has gone a long way towards solving the problem of predicting the number of hosts on a network, it failed to solve the other difficulty presented in Chapter 4 — that is, physically locating the hosts on the network. Chapter 6 will attempt to address this issue by presenting an alternative approach to locating computers within a building.

Chapter 6. Finding Specific Machines

A significant problem facing most network administrators is that of keeping track of where computers are located on a network. This location is both physical and logical — physical as in what room the machine is located in, and logical as in to which switch port a host is connected. This information is part of the configuration management area in the OSI network management model. Knowing the location of a computer is important when trying to diagnose network faults, especially when the fault can be seen to originate from a particular machine.

Traditionally, this problem has been solved by maintaining meticulous records of all movements of computers and changes in network cabling. These records can be refered to in order to determine where a particular network component should be, either logically or physically. Unfortunately, records such as this are seldom adequately maintained. The result is that as networks evolve, less is known about how the various components that form the network are interconnected. This is particularly true of networks that develop in an *ad hoc* manner.

This phenomenon is best illustrated by a news story that appeared on several reputable wire services in April 2001. When an equipment audit was conducted at the University of North Carolina in the United States, it was discovered that one of their Novell servers was missing. While the computer was working and visible on the network, nobody knew where it was physically located. An extensive search revealed that the server in question had been sealed behind a dry wall a number of years earlier [TechWeb, 2001].

While most cases are not as extreme as the one at the University of North Carolina, the problem of locating machines is certainly one that exists in a large number of networks. It is a problem that also exists within Rhodes, so once again, Rhodes provided a useful case study in which to experiment with a solution to this issue.

As was observed in Chapter 5, Hamilton Building (housing the Departments of Computer Science and Information Systems) was constructed in 2001 and fully occupied in January 2002. A new building presents an ideal opportunity for getting things right, and indeed, during the construction phase, detailed records were kept of all network installations. As is typical with a new building, there was a settling in period after the two departments moved in, and during this time many changes were made to the network infrastructure. Records of these changes are sketchy at best, leading to the sort of situation that was described above.

6.1. Logical location

The logical location of a computer can be worked out by tracing network cables back from the computer to the switch that it is connected to. This is a tedious and time consuming way of finding out how a network is interconnected, and ideally would only be done when a new network point is installed. A software solution would be far more convenient, and would allow records to be more easily updated

and maintained.

While software solutions are feasible for managed switching infrastructure, they are impossible for unmanaged network devices. This is because, in an unmanaged network, there is no way of associating a switch port with a computer other than physically checking which cable is plugged into which port. For this reason, this section will only look at managed network infrastructure.

The SNMP MIB-II specifies a way of retrieving an address resolution protocol table from a network device. On an OSI layer two network device, this ARP table contains an association between a network interface controller's MAC address and the interface on the device through which that network interface controller is visible. Entries in this ARP table have a limited lifespan and are refreshed every time traffic is directed at a host that is connected to the device. On a managed network device, an SNMP agent on the device can be used to read this table.

All computers connected to an Ethernet network maintain their own ARP tables. Unlike layer two network devices, devices at layer three or above that use the Internet Protocol (IP) maintain ARP tables associating MAC addresses with IP addresses. This ARP table contains, at a minimum, every host on the local subnet which has recently engaged in communication. Again, ARP entries have a limited lifespan and are refreshed every time traffic is sent to another host. Those hosts that are beyond the subnet boundary do not have an entry in the ARP table — a single entry for the default gateway is maintained instead.

These two bits of information can be combined to provide a correlation between a specific host on the network and a switch port.

In order to ensure that there is a valid entry for the host in the monitoring machine's ARP cache, it needs to generate some traffic to that host. This can be as simple as a single ICMP echo request or ARP whois request to the host. The arpwatch(8) program automates this to a large extent. Once this has been done, the monitoring machine's ARP table can be queried to determine a MAC address for the host. Using the SNMP MIB-II, this MAC address can be looked up in the switch's ARP table to determine to which interface the host is connected.

This works very well for networks where there is a single switching device. Unfortunately, real world networks tend to have significantly more than one switch — Hamilton Building, for instance, has thirteen switches serving over four hundred network points.

The algorithm for determining which interface on which particular switch a host is connected to is a little more complex. First of all, it is necessary to have some idea of the topology of the network. This can either be manually configured or automatically learned (as in Section 4.2).

Given a set of switches serving a particular subnet, the interface that a host is connected to can be determined by first querying any one of the switches to determine on which interface the host's MAC address can be seen. This query will give the interface on the queried switch that serves the host in question. It can be determined from topology information if the interface in question is a trunk port (an uplink or a downlink to another switch) or whether the interface serves the host directly. If the switch

serves the host directly, the search terminates and the interface is known.

If, however, the interface on the queried switch is a trunk, the search moves to the switch providing that trunk. The process of querying the ARP table to determine which interface the MAC address is visible on is repeated on this new switch. This continues until an interface is found that provides direct connectivity to the host in question.

Since this is effectively a search tree, the number of switches that needs to be queried can be minimised by starting at the root of the tree. This root should either be the switch that has the most trunks attached to it or the core switch (usually a layer three switch providing virtual LAN (VLAN) capabilities). Experimentation will show which is the best root of the search for those cases where the switch with the most trunks is not the core switch. This is probably an uncommon situation though.

6.2. Physical location

In Hamilton Building, as with many other installations, each physical network point is labelled with a number that uniquely identifies it. Common numbering schemes either relate the network point to a switch port, or to the room or building in which it is located. The latter of these schemes is the most flexible since it allows physical network points to be re-designated and patched into different switch stacks without leading to inconsistent numbering.

Detailed records need to be kept of the location of network points in order to allow the network administrator to determine, from a network point number or switch port, where in a building a computer is located. The larger and more widespread a network becomes, the more important it is that these records are kept up-to-date.

While it is possible to write a computer program that detects the logical location of a computer, this is not usually possible for the physical location. Traditionally, determining the physical location of a network point has involved someone walking from office to office making a note of the location of each point.

This task can be made a lot simpler if some form of computer assistance can be provided in order to enable the administrator to quickly record the location of the network point. Ideally this should provide the administrator with the ability to enter a network point number and get a floor plan indicating a “best guess” as to where the network point may be. The person using the system should be able to then update the guessed location by simply indicating a more accurate location on the floor plan.

The ability to provide a best guess for where a network point should be is dependent on how much information is contained in the network point’s numbering scheme. If the number refers to a switch port, a best guess could be a building or the portion of a building that the particular switch serves. A numbering scheme that includes an indication of the location of a port, such as the one used in Hamilton Building, provides a more accurate best guess. As much information about the location of ports as is known should be provided beforehand in order to make the best guess as accurate as possible.

With the advent of handheld computers and wireless networking, the network administrator is presented with new tools to aid him in his never-ending quest to ensure that records of network infrastructure are kept up-to-date. A handheld computer (such as Compaq's iPAQ) provides a convenient way to display and update the location of each network point as they are located.

It would be far better, however, if a means could be provided for those people using computers to update their own records. This is the combined approach taken in Section 6.3.

6.3. A combined approach

In order to work out this location information for Hamilton Building, an online database system was constructed. A combined approach was taken, allowing both the physical and logical location to be determined using the same application.

Where possible, this information was to be gathered by the end users of the network, so it was important that any such system be simple and easy to use. To achieve this, a web interface to the database was developed. The interface was designed such that, in general, the user need only enter the number of the network point to which his computer is attached. The rest of the information required to populate the database was, as far as possible, gathered automatically.

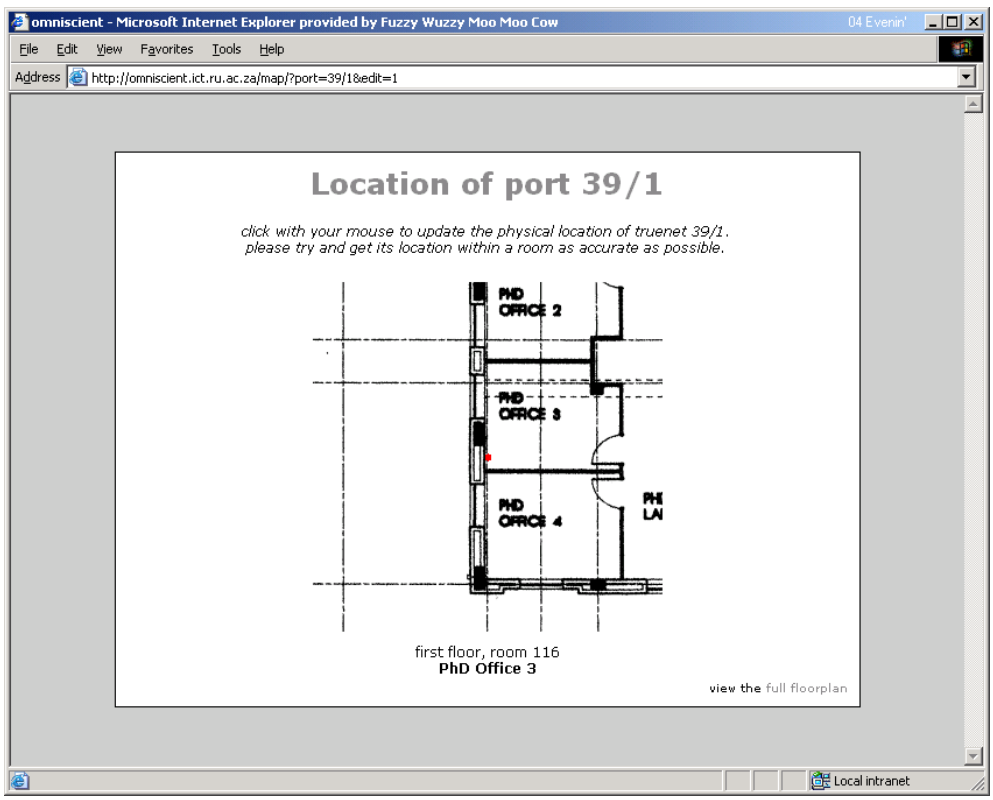
The database running this application keeps two pieces of information. It stores a correlation between a switch port and the network point number (the logical location) and the user-entered coordinates of the network point as represented on a floor plan of the building (the physical location). These tables were populated with as much information as was previously known about the network in order to minimise the amount of information that would have to be collected from the end users.

Fortunately, the point numbering system in Hamilton Building makes it easy to correlate a network point with as specific room in the building. The network point number is divided into two separate parts, for example *37/1*. The first part (*37*) uniquely identifies the room that the point is in and the second part (*1*) identifies a particular port within that room. The room number on the network point does not, however, correspond to the room number on peoples' doors. The relationship between these two numbers is known from the building plans and the database was pre-populated with a relationship between the two. The location of each room on the floor plan was also established and entered into the database. These two bits of information allow the system to make an accurate determination of the physical location of a network point.

When this web application is accessed from any computer in the building, the system first uses SNMP to locate which interface on which switch stack the remote computer is connected. This information is then looked up in the database to determine if there is already a correlation between this interface and an existing network point. If no such relationship is found, the user is prompted to enter the number of the network point to which their computer is attached. This point number is then recorded in the database against the interface number from the switch stack.

Once this information is known, the user is presented with a web page detailing the network point number, the room number and a description of the purpose of the room in which their computer is located. A subsection of the building's floor plan that pinpoints the position of the network point is also displayed. If no location is currently known for the network point, this subsection simply shows the room in which the network point is thought to be located. The user is then asked to more accurately position the point on the plan using their mouse. Any updates the user makes are saved back into the database. Figure 6-1 shows an example page from this system.

Figure 6-1. Location of a network point



6.4. Shortfalls

There were two problems experienced with this system, one technical and one social.

Nortel equipment forms the majority of the switching infrastructure in Hamilton Building. At the core of the network are thirteen Baystack 450 switches stacked together to form three separate switch stacks. Each of these stacks runs its own SNMP agent that serves all the switches in that stack. All three of

the stacks are running identical versions of the Nortel firmware on identical hardware. They are also all configured in a common way.

For some indeterminable reason, the SNMP agent on one of these switch stacks is unstable. It operates perfectly when the stack is first powered on, but over time it starts failing to respond to some SNMP get requests. The original solution to this problem was to simply re-send the request until such time as a valid SNMP response was received. However, the performance degrades over time until it reaches a point where it simply stops responding to queries at all, making this approach useless. It seems that once this has happened, the only solution is to power cycle the switch stack. The other two switch stacks operate entirely as expected. Unfortunately, the stack that displays this erratic behaviour is the stack for which this application would prove most useful — it is the stack that provides connectivity to staff machines.

It should be noted that the interface ARP tables that this application uses come out of MIB-II, which means that they should be standardised across vendor platforms. As such, the application should not differentiate between different makes and models of devices. Nevertheless, Section 3.2 provides a solution to any vendor-specific quirks should they arise.

Socially, it is difficult to convince the end users of the need to gather this information. Although it takes under two minutes to complete the process, a test run with some post graduate students showed that less than a third of the people asked actually took the time to enter their network point number. This issue can be solved to a large extent by user education.

The bigger problem is that the students who did take the time to complete the process were concerned about the amount of information that was known about their location prior to their completing the form. In other words, the pre-population of the database gave people the impression of a “big brother” type application. This feeling of being watched may explain why other people were reluctant to complete the process.

There are two different solutions to this issue. The first is to simply not pre-populate the database. This would require that people entered more information about their computer and increase the amount of time taken to gather the data (especially from shared workspaces where there is common information between a number of end users). The second is, once again, user education. The process of pre-population needs to be carefully explained to people in such a way that they understand that its function is to make it quicker and easier for them to enter the specific details of their machines.

6.5. Summary

This chapter looked at a configuration management application that fulfilled some of the issues raised in Chapter 4. The importance of knowing both the physical and logical locations of each host on the network was explained and an application was developed in order to assist with the determining of this information.

While this application was developed to solve a specific problem, it could easily be used in any other place that shares a similar port numbering scheme to Hamilton Building. Some minor changes would allow it to be used with other numbering schemes. As was noted earlier, the method described in Section 3.2 could be used to overcome any problems with vendor interoperability.

Chapter 7. Intelligent Reporting

Part of the fault management section of the OSI network model covers the timeous reporting of faults and error conditions to the network administrators. Such reports are necessary in order to allow network administrators to quickly respond to and diagnose problems. The more accurate the information that is provided to the administrator, the better their ability to respond. Unfortunately many existing network monitoring systems (such as Big Brother, which was mentioned in Section 2.2.6) are limited in their ability to provide fault reports — they report on the symptoms of the problem rather than the cause of it.

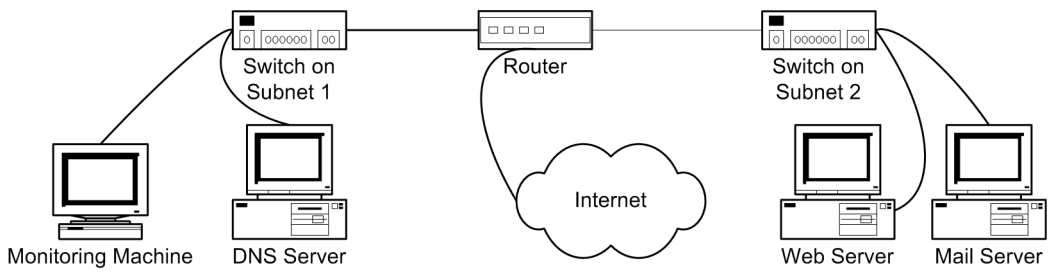
It would be useful if this limitation could be overcome and more intelligent reporting systems could be developed. These systems could make use of various forms of artificial intelligence in order to make their own diagnosis of faults rather than relying on the network administrators to perform this tedious task. Such intelligent systems form the focus of this chapter.

7.1. Symptomatic reporting

Big Brother provides a good example of typical traditional network monitoring utilities. It can be configured to periodically connect to various network services, and check that these services are still functioning as intended. It has various notification abilities and can be configured to alert the network manager if these services are not working as they are supposed to [BigBrother, 2002].

The biggest limitation of systems such as this is that they can only symptomatically report error conditions, and the symptoms are limited to those services that have been configured to monitor. The best way to illustrate this is via an example. Consider the network in Figure 7-1.

Figure 7-1. A simple example network



In this network there are two completely separate subnets that are interconnected by a layer three router (this could also be a layer three core switch). On one subnet there is a DNS server and the machine

responsible for network monitoring. On the other subnet there is a web server and a mail server. The router in between the two subnets also knows how to route packets to the Internet.

The network monitoring station is configured to check that the web server is correctly functioning as a web server, the mail server is correctly functioning as a mail server, *et cetera*. It does this by, for example, connecting to the web server and trying to request a web page. Should any of these services go down, it will report the issue to the network administrator in some way (the actual method of reporting is not significant to this example).

This works very well in many cases. If the web server ceases to function, the monitoring application will pick this problem up and report it.

What happens, however, when the switch serving subnet 2 stops passing packets? The monitoring machine attempts to contact the web server and fails because it is no longer reachable. It reports this matter to the network administrator. Shortly afterwards, it attempts to contact the mail server and discovers that it too is not accessible. It sends another report to the network administrator alerting them of this fact. In both cases, the reported faults are symptoms of the real fault. It is up to the network administrator to figure out where the real fault lies.

Now consider what happens when the subnet 2 supports a large number of servers, each monitored, each performing a different function. The network administrator may be overwhelmed with problem reports.

In large networks where there is a distribution of responsibility, these symptomatic fault reports can be even more problematic. Say for example, that the web server is managed by the webmaster [1] and the mail server is managed by the postmaster (and these are two different people). When switch 2 fails, the webmaster will receive a report about the web server and the postmaster will receive a report about the mail server. To continue the example, say the network infrastructure is managed by a third person, the hostmaster. In this case, the hostmaster receives no reports of any faults and relies instead on the postmaster and webmaster informing him of the problem.

It is fairly easy to see from these simple examples why symptomatic reporting is problematic. As the complexity of the network increases, these problems can only be compounded. What is needed to improve the situation is some form of intelligence within the network monitoring software — it needs to be able to diagnose simple network faults by itself and report them more appropriately to a person who is in a position to resolve the problems.

7.2. Expert systems

A lot of research has gone into the subject of artificial intelligence in computing and a discussion of the merits of various approaches to solving this problem is beyond the scope of this work. This section

1. The names “webmaster”, “postmaster” and “hostmaster” are derived from RFC 2142, Mailbox Names for Common Services, Roles and Functions.

presents a very brief overview of the common methods by which artificial intelligence is implemented in order to provide some of the background for Section 7.3.

7.2.1. Rule based systems

The simplest of the artificial intelligence approaches is that of rule based systems. These systems attempt to mimic the logical steps that go into solving a specific task. They have a predefined set of questions and a set of options that form the answer to each question. As each question is answered, a set of rules is applied to the answer in order to determine what the next step will be [Patterson, 1990].

Rule based systems have no ability to automatically learn from their mistakes, nor do they have any way of determining information from their environment. As such, their use is usually limited to very simple problems that have a finite, known set of possible states.

7.2.2. Expert systems

Expert systems provide a way of using the knowledge of one or more experts in a field to solve a problem in a particular domain. Each expert system has a limited amount of knowledge derived from these experts which make it, in turn, an expert in the same field as the experts from which it obtained its knowledge. Expert systems are based on the rule based systems described in the previous section with one notable exception: the set of rules that an expert system applies are known as heuristic rules — that is, they are based on knowledge known to the collective expertise in that domain rather than simple facts that are accepted as common knowledge [Levine, 1990].

As they derive from rule based systems, expert systems suffer from some of the same limitations as their simpler relatives. It is generally quite difficult to expand their knowledge base, for example. They are also limited to problems that have clear cut, “black and white” solutions, rather than problems for which there are a number of possible intermediate solutions. Expert systems are, however, able to solve complex problems in the domain for which they have expert knowledge.

7.2.3. Pattern recognition

Ford argues that from a philosophical point of view, one cannot have true intelligence until one implements some form of pattern recognition [Ford, 1987]. Thus he dismisses the rule based approaches to artificial intelligence as being commercial oddities and not real intelligence at all. In many ways he has a point; most real world problems are not particularly clear cut, and the use of pattern recognition is an attempt to solve this.

Rather than applying a predefined set of rules, systems employing pattern recognition take all the information they have been supplied and attempt to match it to information that they already know. In

this way, a best-match approach can be taken to try and solve problems for which there is no exact information available.

Systems that employ pattern recognition techniques tend to be much better at solving problems for which they have existing information than either expert systems or rule based systems. Like their predecessors, however, they lack the ability to easily learn new information.

7.2.4. Neural networks

Neural networks, as their name suggests, attempt to mimic the way the human brain operates. They provide intelligence in the conventional sense of the word, by combining sophisticated pattern matching with the ability to learn from their mistakes and apply new knowledge. Over time, neural networks become experts in the field in which they have been trained — that is, they apply past knowledge to new problems in an attempt to find a solution.

While they are good at solving complex, “fuzzy” problems, neural networks are a lot more complex to implement than any of previous approaches to artificial intelligence. Part of the reason for this is that neural networks need to be trained rather than just acquiring a predefined set of knowledge.

Neural networks are often used to solve complex problems that are beyond the realm of human understanding, for example, making accurate predictions of weather systems. They are good at providing solutions to these problems because there are vast amounts of historical data available from which to aquire a knowledge base.

7.3. Intelligent network monitoring

It seems that almost all active network monitoring tools that are currently available are, at best, rule based systems. This lack of expert information or “intelligence” results in the symptomatic reporting described in Section 7.1. To improve this situation, it is necessary to provide network monitoring agents with the ability to *interpret* the information obtained from scanning the network.

7.3.1. Gathering data

As the first approach to this problem pattern recognition techniques were employed. For example, attempts were made to write software that determined if a particular sequence of events preceeded a network failure. The intention was to progress from these techniques to full artificial intelligence, of the sort provided by the neural network approach.

While attempting to obtain a knowledge base for this system, however, it was discovered that diagnosing the vast majority of network faults is a fairly straightforward, logical process. The “grey areas” commonly associated with neural networks simple do not exist for the bulk of the faults such a system is likely to encounter.

Part of the reason for this is that, in general, the sorts of probes done by networking monitoring have a binary outcome — the results are clear cut, the probe either succeeded or failed. For example, if an ICMP echo request is sent to a machine to determine if it is actively participating in the network, the machine will either respond with an ICMP echo response or it will not respond at all. The protocol does not allow for any other responses to be valid.

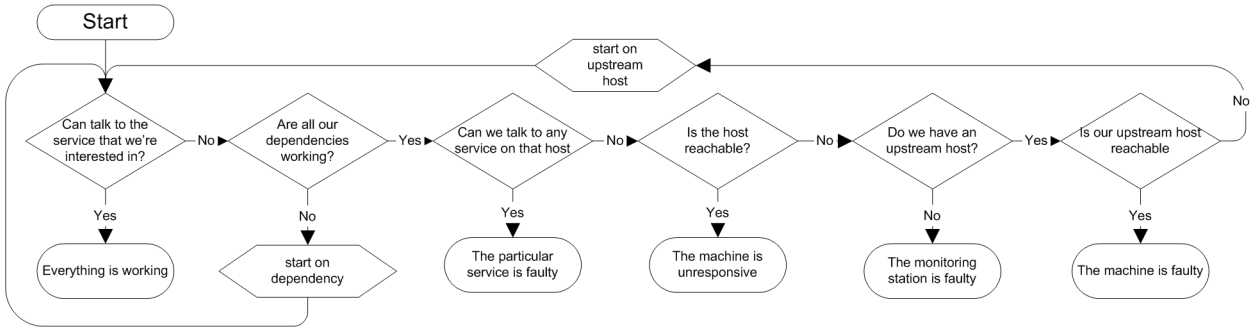
This sort of binary logic lends itself to a decision tree or graph based approach, where each node in the graph is characterised by some form of “intelligent” question. In other words, the system requires the type of expert knowledge mentioned in Section 7.2.2. Thus the idea of a pattern based system was discarded in favour of a simpler expert system.

The most important piece of knowledge required by an expert system doing network monitoring is knowledge of the layer two and layer three topology of the network being monitored. In particular, the monitoring machine needs to know the normal route taken by packets between itself and the machine or service it is monitoring. Chapter 4 discusses methods for obtaining this information at both layer two and three of the OSI network stack.

Knowledge of the dependencies between various computers is another useful piece of information that can be supplied to expert network monitoring systems. To take the example presented in Figure 7-1, the mail server relies on the DNS server in order to be able to deliver outgoing mail (since it has to perform a DNS lookup in order to determine the IP address of the machine it wishes to deliver mail to). If the DNS server is down, the mail server must necessarily be running in a degraded state.

Using this information, a network-specific expert system was constructed. The set of rules for such a system turns out to be remarkably simple. Part of the reason for this is that the same set of rules gets applied to each host that is queried, resulting in multiple recursive calls to the same set of of rules. Figure 7-2 shows just such a recursive set of rules forming the basis of an expert network monitoring system.

Figure 7-2. Decision process for determining network faults



As can be seen from the flow chart, the monitoring of any specific service begins with a check to see whether that service is running on the machine that it is supposed to be running on, and whether it is performing as expected. At any stage during the tests a conclusion can be reached, depending on the results of the specific test that was run.

If the service is faulty, any dependencies of that service (for example, DNS in the case of mail) should be checked. Provided all the dependencies are working correctly, a check should be made on any other services that are known to be running on the machine. This will determine whether it is a specific service on the machine that is faulty, or if there is a more general problem.

Assuming that no services can be reached, a test should be performed on the reachability of the host in general — this is simply an ICMP echo request to determine if the machine is available on the network. It cannot be assumed at this point that the machine itself is faulty (this is the mistake many current network monitoring tools make).

Before that can happen, any upstream hosts that provide connectivity between the monitoring station and the host being monitored need to be tested. As was shown in Section 4.1, there is always at least one hop between any two machines (the network interface on the machine receiving the packet). If there is more than one hop, each of these hops needs to be tested in turn before a conclusion can be drawn about where the fault lies.

The process for testing each of these upstream hosts (and each of the dependencies for that matter) is identical to the one outlined above, which is what makes this approach to testing recursive.

In practice, implementing this recursive expert system proved to be challenging. The main problem that needed to be addressed was a method for deciding when recursion should stop. While this may seem straightforward from the decision tree in Figure 7-2, real network topologies with redundant routing create loops. These loops need to be detected and dealt with in such a way that the system does not recurse *ad infinitum*.

Once these diffculties were overcome, a proof-of-concept implementation of the decision tree represented by Figure 7-2 showed that such a system could, for all the test cases at least, accurately diagnose

faults on the network.

7.3.2. Testing services

One of the tests mentioned in Section 7.3.1 involves connecting to a service and testing that it functions correctly. This is perhaps the most difficult task this system has to perform, since accurately testing a network service involves an understanding of the underlying protocol that the service uses.

Supporting the commonly used protocols such as those used by e-mail, web pages, *et cetera* is a fairly straightforward task. The protocols are well understood and well documented, so all that is required is a minimal implementation of the client side of the protocol. The problem arises when trying to support the less common protocols.

One method for doing this is to simply provide a basic connect test for those protocols which have not explicitly been covered. In this test, being able to connect to the port running the service is assumed to be sufficient — the service is considered to work in this case.

This is the approach taken by the proof-of-concept system that was developed to test these ideas. A services test module was implemented in Perl, and this module was called by the expert system to test the availability of services. Test subroutines were named after the protocol's assigned port keyword, as maintained by the Internet Assigned Numbers Authority (IANA) [RFC 3232]. Examples of such keywords would be "telnet" for the Telnet protocol, "http" for the HTTP protocol, and "domain" for the DNS protocol.

Any protocol which does not have an explicitly specified test routine is handled by Perl's special `AUTOLOAD` sub-routine, which allows programs to simulate the existence of missing sub-routines. When this routine is called by the test module, it uses the name by which it was invoked (in other words, the name of the missing sub-routine) as an index into the services(5) database, from where it obtains the IANA-assigned port number for the service. The system then attempts to connect to this port on the monitored host in order to determine whether any service is listening on that port.

As has already been mentioned, simply connecting to the port is not a particularly accurate way of determining whether a service is functioning correctly. A better approach would be for the system to learn new protocols automatically, and in that way compare the results of previous tests against the current one.

The problem with this approach is that most protocols allow implementors some latitude in the way their services announce themselves or respond to queries. For example, Figure 7-3 shows the welcome banner from four different Internet Message Access Protocol (IMAP) servers. The IMAP protocol requires that an IMAP server identifies itself on connection with `*OK`. The rest of the line is ignored by any client, and is simply provided for informational purposes [RFC 2060].

Figure 7-3. Various IMAP implementations

```
* OK Courier-IMAP ready. Copyright 1998-2002 Double Precision, Inc.  See COPYING for distribution information.

* OK imap.ru.ac.za Cyrus IMAP4 v2.1.9 server ready

* OK Microsoft Exchange IMAP4rev1 server version 5.5.2653.23 (stork.ict.ru.ac.za) ready

* OK GroupWise IMAP4rev1 Server Ready
```

Any application that attempts to automatically learn, for example, the IMAP protocol would need to recognise that it is only the `*OK` that is important, and discard all other information. This problem is a “fuzzy” one, and is perhaps best suited to higher forms of artificial intelligence, such as a neural network.

While this approach seems to be the logical extension of the system currently employed to test services, due to time constraints, an implementation was not forthcoming. It remains to be seen if a neural network could be used to accurately determine whether services are functioning correctly, and indeed, if the results from such a system would be better than the straightforward connection approach currently employed.

7.3.3. Reporting faults

It is all very well being able to accurately determine what the cause of a network fault is, but without the ability to report the fault to a relevant person, the process is futile. Thus it is important that an intelligent method of reporting network faults is developed alongside the intelligent monitoring system.

Specifically, this reporting system needs to be able to determine who needs to be made aware of a particular fault.

In order to do this, a reporting system needs to have knowledge of which people are responsible for which services and areas of the network, and their preferred method of communication. In addition, the system needs to know the dependencies between various services, in order that it might be able to determine which other operational areas may be affected by the fault.

Two of the major shortfalls of the current symptomatic reporting are that administrators receive reports for faults outside their area of operational control, and they often receive multiple, different reports for the same fault. These problems can be solved by the use of a rule based or expert reporting system. Such a system should necessarily be fed information by an intelligent monitoring system.

The system was developed to report such faults used the Short Message Service available on GSM cellular networks in order to notify the responsible people of the existence of faults on the network. (The method by which such reports were made is discussed in Appendix A.) Reports were divided into two categories: those that were purely informational, and those which required some action on the part of the recipient.

When a particular service failed, a message was sent to the administrator of the service in question. This message explained in as much detail as possible the location and nature of the fault. It also listed the dependencies of the service in question so that the administrator could make a judgement call on how urgently the problem needed resolving. An example of a message that could be generated by this system is "`sws-pgrad.ict failed (ping,telnet,snmp), rm 42, depends diablo.ict,sears.ict (ssh,smtp,http,netbios-ns)`".

At the same time, an informational message was sent to the operators of each of the dependencies informing them of the failed service. This message was not intended to invoke any response on their part, but rather to keep them aware of the fact that their service was running in a degraded state. Once a fault was resolved, a further informational message was sent to each dependency administrator informing them of the return to normality.

The system was designed in such a way that no one person received more than one message alerting them to a particular problem. To take the example in Figure 7-1 and assume the webmaster and the hostmaster are the same person (that is, the person who runs the web server also runs the DNS server). If the DNS server was to fail, the system would notify the hostmaster of the failure. DNS is a dependency of the web server, so the webmaster should receive an informational message informing them of the DNS fault. Since the responsible person is the same, however, this message will never be sent. Should there be more than one responsible person for the web server, only those people who have not been notified already will be sent an informational message.

In the same way, the system was designed so that at any one time, no person should receive more than one message. Carrying on with the example from the previous paragraph, take the case where the webmaster and the postmaster are the same person (that is, the person who runs the web server also runs the mail server, but not the DNS server). In this case, there are two dependencies about which this person needs to be informed when the DNS server fails. The system will concatenate these messages into a single message containing information about both dependencies.

The reason for this policy of never sending more than one message to any particular operator is a simple social engineering technique. Experience with other notification systems dictates that when people receive multiple messages notifying them of a problem, they tend to ignore all but the first. If this first message contains all the appropriate information, there is a better chance that it will actually be read and acted upon.

7.4. Summary

This chapter looked at the OSI fault management requirement for the reporting of network faults. It examined the traditional symptomatic methods for doing this, and commented on some of their shortfalls. The idea of intelligent systems was introduced, and their use in creating intelligent network monitoring utilities was discussed.

The need for the topology information obtained in Chapter 4 as a pre-requisite knowledge base for such an intelligent system was discussed, as was the recursive nature employed by systems aimed at detecting network faults. The testing of various services was examined, and it was pointed out that this may be a good application for neural networks.

Fault reporting was also discussed, particularly the need to generate concise, accurate fault reports. A strategy for achieving this through the use of an expert system was introduced.

Chapter 8. A Coalescence of Systems

Several different network monitoring solutions have been examined thus far, each of which was designed to solve a specific problem. In turn, each of these applications has addressed one or more of conceptual areas identified by the Open Systems Interconnect network management model.

This chapter seeks to clarify how each of the applications that have been discussed fit into the OSI model, and how the various applications depend on, and interrelate with each other in order to paint a broader picture of the work that has been covered.

Some of the systems presented in earlier chapters were designed to address specific problems at Rhodes University. In all cases, however, the concepts behind each of the applications are applicable to a large variety of other situations, as will be shown later in this chapter.

8.1. Open systems interconnect model

The OSI model for network management was presented in Chapter 2 as a framework into which the goals of network monitoring could be fitted. The five conceptual areas of network management are performance management, configuration management, accounting management, fault management and security management. Every network monitoring application fulfils some or all of the roles associated with one or more of these network management areas.

It is important to understand the intended role of each network monitoring application, and the OSI model provides a useful way to outline the tasks a particular system is intending to achieve. The applications described in earlier chapters of this work each relate to specific areas of the OSI model, and will now be examined.

8.1.1. Performance management

Three of the applications developed during the course of this project fall under the banner of performance management: the RADSL monitoring mentioned in Section 3.1, the tracking of network growth that was the subject of Chapter 5 and the logical location of machines, as described in Section 6.1.

The first of these, RADSL monitoring, is perhaps the most relevant application of performance management. Detailed records of the performance of each of the forty-four remote access lines terminating in Hamilton Building are kept, with historical information for up to a year. This information is used to determine when the performance of these lines falls below acceptable limits, or when there is a significant change in the performance of the lines.

Determining the logical location of machines also comes under the heading of performance management, since the positioning of elements within a network directly has an impact on the performance

of the network — for example, highly utilised servers should be positioned at the core of the network rather than on low bandwidth leaf nodes. By tracking the logical location of hosts on the network, the impact of changes on the performance of the network can be predicted.

The last of the three applications, tracking and predicting network growth, is in itself a goal of performance management.

8.1.2. Configuration management

The area of configuration management is vast and includes most of the applications that have been discussed in this work to some extent or another. Specifically, the network maps described in Chapter 4, the tracking of network growth from Chapter 5, and the location of machines that was discussed in Chapter 6 all fall under the umbrella that is configuration management.

The layer two and layer three network maps that are presented in Chapter 4 can be used to monitor changes in the configuration of a network, as can the location applications presented in Chapter 5. These changes could be either intentional or unintentional. In the case of the latter, historical information kept about the configuration of the network can be used to restore the network to its normal state, as well as to help track down the cause of the change.

The application presented in Chapter 5, on the other hand, provides a way to predict and pre-empt the need for changes in the configuration of the network. For example, if a sudden growth is experienced in a particular part of the network, the network can be reconfigured to provide the necessary infrastructure to cater for this growth.

8.1.3. Accounting management

While none of the systems described in this work were directly intended to provide any sort of accounting management capabilities, any application that keeps track of the addition or removal of hosts from a network can be used to feed information into an accounting system.

Say, for example, that a particular organisation has a policy which required that entities (people or departments) are billed for each connection to the organisation's network. Any application that detects the presence of hosts on the network could contribute useful information about when and where a host was connected to the network, and how long it remained connected.

In the same way, applications that monitor the presence of machines on a network can be used to enforce policies regarding the access and use of that network.

8.1.4. Fault management

The area that is perhaps most conventionally thought of when people envision networking monitoring is that of fault management. It is not surprising, therefore, that fault management forms a large part of the systems that have been examined in previous chapters, with each system fulfilling some of the criteria set out by this conceptual area.

Intelligent reporting of faults, as presented in Chapter 7, is almost exclusively a fault management system — its sole function is to detect the occurrence of faults on the network and report them, in an intelligent manner, to the network administrator.

Other systems that feature a strong fault management component are the RADSL monitoring in Section 3.1, the network maps of Chapter 4 and the locating of machines in Chapter 6. Each of these systems could, in turn, be used to feed the intelligent reporting component of Section 7.3.3.

8.1.5. Security management

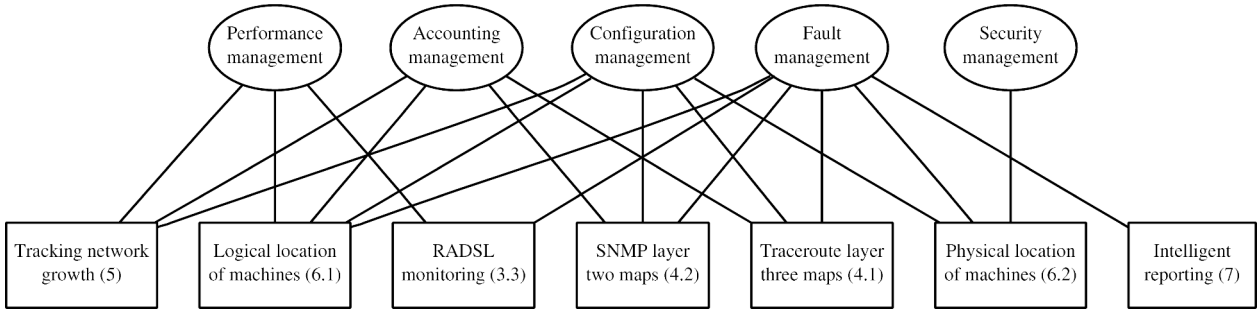
Only one of the systems examined thus far falls directly into the area of security management; the physical location of machines described in Section 6.2 can be used to determine whether machines that contain confidential or other sensitive information are located in an area that is physically secure.

That said, data obtained from other systems, such as the logical location of hosts from Section 6.1 could be used to feed systems in the security management area with relevant information. In the same way, systems that plot maps of the network topology, like those discussed in Chapter 4 can be used to determine which intermediary hosts (routers, *et cetera*) could be considered a risk to the security of a particular machine.

8.1.6. OSI summary

Figure 8-1 provides a summary of how the various systems and components that have been discussed in the section relate to the conceptual areas of the OSI network model. In this figure, the rectangular boxes represent systems that have been discussed (the figure in parenthesis is the chapter or section number that describes the system in question), and the ovals represent OSI conceptual areas. The lines between the two show which conceptual areas each system falls into.

Figure 8-1. Relationships to OSI network management areas



The most notable feature of Figure 8-1 is the number of interconnections that it shows, indicating the coverage of a large number of conceptual areas. The only exception is the OSI security management, which stands out on its own.

8.2. Dependencies

A number of the systems that have been examined in this work depend on other systems that were developed during the course of this project for information and configuration data. This section aims to make these relationships clear, in order to explain how the systems function together to form a synergistic network monitoring system.

8.2.1. Dependencies between systems

The fundamental basis of a large number of the applications that have been described is the XML-based abstraction layer that was introduced in Section 3.2. This layer allows applications to operate independently of the network infrastructure by providing a way of abstracting from the vendor-specific components of the infrastructure in question. Having an abstraction layer such as this also makes the systems developed on top of it highly portable — all that needs to be changed in order to get the system to function on a new network is the abstraction layer's configuration.

The RADSL monitoring system of Section 3.1 is a prime example of why having such an abstraction layer is useful. The asymmetric nature of the RADSL line speeds, and the corresponding enterprise extensions to the simple network management protocol's MIB-II are the main reason that this XML-based abstraction was developed. As a result of the abstraction layer, the RADSL monitoring system makes no distinction between information that is derived directly from the SNMP MIB-II and that which is obtained from the enterprises section of the MIB. This means that the system can be used to monitor RADSL systems made by other vendors.

For the same reason, generating layer two network maps using SNMP (as was done in Section 4.2) benefits from the use of an abstraction layer. In this application, the XML middleware layer does more than abstracting from the vendor-specific components of the network — it is used to provide some seed information about the topology of the network in order that the application can decide where to start its mapping procedure.

Working out the logical location of a computer, as was done in Section 6.1 relies on both the XML abstraction layer directly, and on the layer two network maps that are obtained via SNMP — meaning that this application has a twofold reliance on the XML abstraction layer. It uses the network maps to provide seed information about the location of the various layer two switching devices on the network, and the XML abstraction layer to determine how to communicate with the SNMP agent on each device. In addition, this application depends on the traceroute(8)-based layer three network maps described in Section 4.1.

Chapter 5 describes an application aimed at tracking network growth. This application uses ICMP echo requests to determine whether there is a host listening on each IP address within a particular network block. In addition, the system can used to record the route taken by packets between the monitoring machine and the host in question. This route is recorded in much the same way as the traceroute(8) program records a route. In this way, it is a direct extension of the layer three network maps that were discussed in Section 4.1 The network mapping system records the hops between two hosts, in other words the intervening network infrastructure. In the network growth application, this is taken one hop further and includes the end node as well.

Another application that depends on knowing the logical location of machines is the system that records the physical location of the machines. A combined approach to determining location was taken by Section 6.3. This approach collates the information obtained about the physical and logical locations of machines. The result of this unity of approaches is an application that provides a more complete picture of the structure and location of machines on a network.

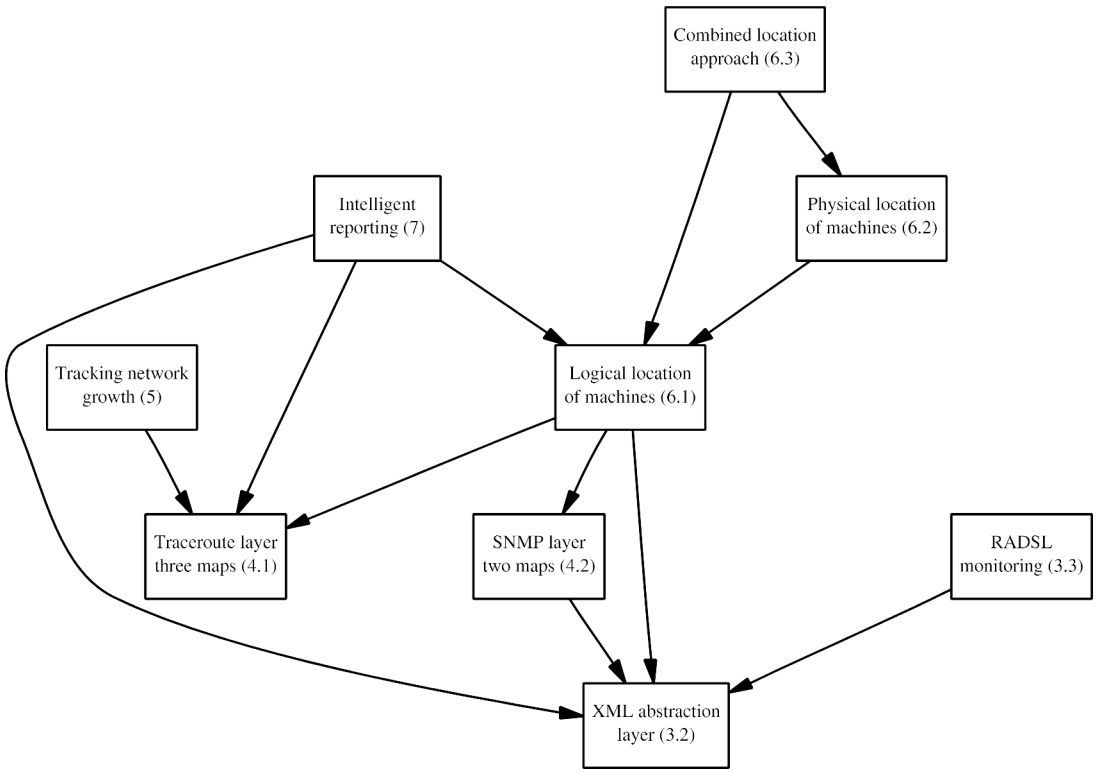
The intelligent reporting of faults, as outlined in Chapter 7 requires knowledge of the dependencies between various machines. These relationships can, to some extent, be determined automatically using the approaches described in Chapter 4 and Section 6.1 — that is, the network mapping and logical location applications respectively. In addition, this reporting application could be supplied with hints on the network topology directly using, for example, the XML-based approach taken in Section 3.2

8.2.2. Dependency summary

Figure 8-2 summarises the relationships and dependencies that are explained in Section 8.2.1. In this figure, the arrowheads indicate the direction of the dependency, with the head of the arrow pointing at the system that is depended upon. The numbers in parenthesis indicate the chapter or section number that discusses the system in question. Only those relationships that have been explicitly discussed are shown — there may be implicit relationships that are not shown. These implicit relationships occur

when applications share common ideas and approaches, such as the use of the traceroute(8) approach shared by Section 4.1 and Chapter 5.

Figure 8-2. Dependencies between different systems



8.3. Portability

As was noted in the introduction to this chapter, a number of the systems presented in the work were intended to solve specific problems at Rhodes University.

Conceptually, however, all of the systems that were developed as part of this project have application outside of the Rhodes environment. All of the problems that these programs have attempted to solve are not unique to the University, but rather represent a spectrum of the problems encountered by network administrators world-wide. For example, as has previously been mentioned, Pick ’n Pay, a large South African supermarket chain, suffers from the same problem as Hamilton Building at Rhodes — determining the logical location of machines on their network.

For this reason, in order to ensure that the applications developed as part of this project would be applicable to a wider audience, care was taken to ensure that they were as portable as possible.

In most instances, systems can be ported by simply altering a configuration file or configuration variables in the source code of the application. For example, the network growth tracking application presented in Chapter 5 simply has a variable that identifies Rhodes' network as a CIDR network block, in other words `$network=146.231.0.0/16`. Changing this network block changes the network that is monitored. In fact, it is possible for the system to monitor networks that are not part of its local area network, although this application is not recommended.

A few of the applications are significantly more complicated to port to other networks; the RADSL monitoring system presented in Section 3.2 and the location monitoring application described in Chapter 6 fall into this category. This section aims to document the issues associated with porting these applications in order to facilitate an easier transfer to other networks. In cases where an application is not specifically mentioned, it can be assumed that porting it to another network is a straightforward task.

8.3.1. RADSL monitoring

The XML-based abstraction layer described in Section 3.2 addressed many of the portability concerns associated with this application. It is not, however, completely devoid of issues as a result of this.

While this application makes no assumptions about how to communicate with the RADSL DSLAMs it is monitoring, it does make assumptions about how these devices are connected to the network. Specifically, the application has knowledge of the VLAN configuration on each device, as well as the fact that there are two manageable devices that each use a slightly different communications protocol.

It is possible to correct these assumptions by extending the XML DTD that was presented in Section 3.2. It was always intended that this would be done, but unfortunately making this correction would involve a significant number of alterations to the existing code base. As it is, the system has been functioning in its current format for just under two years, and for this reason it was felt that the changes were not justifiable.

8.3.2. SNMP approach to mapping networks

The biggest problem with porting this application is obtaining the seed data that it requires in order to be able to accurately map a network at layer two. The system needs to be able to distinguish between hosts and switching infrastructure on a network, and in order to do this, it compares the MAC addresses obtained from the system's ARP table with a list of MAC addresses known to belong to switching infrastructure.

In places where the MAC address of every layer two networking device is known (such as those places that use DHCP to configure these devices), this information should be easy to obtain. Networks, however, tend to grow in a *ad-hoc* manner and, for this reason, information on the network infrastructure is often difficult to obtain.

Chapter 4 discussed an alternative approach to this problem that determined whether or not the device was part of the network infrastructure based on the organisationally unique identifier (OUI), the first three bytes of its MAC address. This approach was based on the assumption that, traditionally, network infrastructure uses different OUIs to network interface controllers, even when they shared a common manufacturer — for example, 3COM manufacture both network interface controllers and switches, but use, among others, the OUI `00:50:04` for NICs and the OUI `08:00:4E` for switches. This approach, however, was never implemented.

8.3.3. Location of machines

Ironically, this is perhaps the most location-specific application that was developed. This is because it attempts to display the physical location of machines graphically by displaying their position on a building floor plan. In addition to high-resolution images of the building's floor plans, this system needs to be pre-configured with a large amount of information concerning the layout of the building before it can be used.

It is assumed throughout the combined location application discussed in Section 6.3 that the network point numbering scheme used in Hamilton Building is the *de facto* way of numbering ports — that is, all rooms in a building are assigned a number, and all ports within that room are assigned a unique number within that room. For example, *35/1* would be the first network point in room thirty-five. The application is not easily portable to any network where this numbering scheme does not hold true.

One of the pieces of information required during the pre-configuration phase of this application is a set of co-ordinates for every room within the area to be monitored. This is time-consuming to obtain, especially where the number of rooms is large. In addition, the system maintains a textual description of the function of each room, which needs to be entered along with the co-ordinates.

As was noted in Section 6.4, this system also has a number of shortfalls associated with it. Most notable of these is the problem of convincing people to participate in the process. This problem will need to be overcome in order to make this location monitoring system useful.

8.3.4. Intelligent reporting

There are three issues relating to porting the intelligent reporting system: the importation of topology information, the configuration of areas of responsibility, and the definition of a reporting system.

Topology information is important to this system since it allows the system to determine where the fault is located. This layout information can either be entered by hand, or can be obtained from the network mapping systems described in Chapter 4.

Areas of responsibility are used by the reporting system to determine which people to notify in the event of a fault. This information is contained in a configuration file, and needs to be entered before the system can be used. In some instances where these areas of responsibility are not already clear cut, this may be problematic.

The definition of a reporting system is perhaps the most important part of the intelligent reporting application. Without this system, the application has no way of alerting administrators of faults and issues that arise. The reporting system currently in use is based on the GSM short message service, but this is not the only method that is available. Custom reporting modules can be written and integrated with the system, ensuring that almost any method of reporting can be accommodated.

Section 7.3.2 describes the method employed by the intelligent reporting system to test services. As was mentioned previously, only tests for the most common protocols and services have been implemented. The administrator of any network that runs services other than those already covered by the testing module may wish to develop further protocol routines for this module. This requires the writing of code to handle the specific requirements of the protocol.

8.4. Existing solutions

There are three fundamental differences between the systems examined in this work and traditional network monitoring systems.

Most network monitoring solutions try to solve the most general case to increase their usefulness and hence their marketability. This makes sound business sense, but this generalisation is often done at the expense of little details. The result is often a system that *almost* does what is required of it, but is not quite correct. This is the major difference between an off the shelf network management package and a purpose-developed system — an off the shelf solution does most of what is required, and a number of other things at the same time; a purpose built solution does exactly what one wants it to do, and nothing more. The disadvantage of purpose-developed solutions is that they are not usually portable to other situations.

The four applications developed as part of this project fall somewhere between these two extremes. They are network-independent enough to allow them to be usefully applied to other situations, but they attempt to solve a very specific problem rather than the general case. This network independence is achieved partly through the use of an abstraction layer that provides vendor independence.

Vendor independence is another difference between the systems developed in this project and the vast majority of commercial network monitoring applications. Most commercial applications are limited to

a single manufacturer's products (or at best, a small subset of manufacturers). The solutions presented in this work are, on the other hand, almost completely vendor independent, as is noted in Section 8.3.

The third difference between these solutions and traditional network monitoring approaches is that, to a large extent, the solutions presented in this work attempt to answer the question at hand, rather than provide the network administrator with the information required to answer the question. This difference is perhaps best illustrated by the intelligent reporting system presented in Chapter 7. Traditional approaches to reporting provide symptomatic information, whereas by employing artificial intelligence techniques, the system in Section 7.3 provides information on the cause of the problem.

To a large extent, these differences are the defining feature of the solutions presented in this work, and are what make the approaches used in this project novel.

8.5. Summary

This chapter has looked at how the various applications in this project fit together to form a synergistic whole; it has looked at the "big picture". Each application was examined in relation to the OSI network management model, with emphasis on how the various applications fulfilled various conceptual areas within the model.

The relationships and dependencies between the various applications were discussed, paying particular attention to those applications that had strong relationships, as well as those that were reliant on information supplied by another system.

The issue of porting these systems was also discussed. While it was noted that applications were designed with portability in mind, it was pointed out that this did not mean that they were free of portability issues. Those applications that had specific portability problems were examined in detail, and the various issues associated with porting them were highlighted.

In addition, the differences between traditional network monitoring approaches and the applications that make up this project were discussed.

Chapter 9. Conclusion and Future Work

As was noted in the introduction, this project set out to discover improved approaches to solving several common network management shortfalls, using the Rhodes University network as a test bed. This chapter seeks to summarise the work that has been done, and to present areas that are open to further research and development.

9.1. Summary of Work Covered

The field of network management was explored in the opening chapters of this work, with particular emphasis on the shortfalls of current monitoring techniques. One shortfall in particular, the monitoring of multi-vendor networks was examined in some detail.

A solution to the problem of multi-vendor networks was proposed; using an XML-based markup language it is possible to provide a layer of abstraction that removes the vendor-specific components of network monitoring protocols such as SNMP. This approach is used throughout the rest of the work to provide independence from the specific implementation of the test network.

Four common network monitoring shortfalls were identified, and each of these areas was examined in detail. In all cases, a solution to the problem was proposed, and the merits and shortfalls of these solutions were discussed. The four areas identified were: the problem of working out the topology of a network, both at layer two and layer three of the OSI reference model; the problem of tracking the growth of networks, and of accurately predicting future trends; the problem of locating specific hosts on a network, both at a logical and a physical level; and the problem of symptomatic fault reporting, and the need for this reporting to be more relevant.

Network maps were drawn at both layer two and layer three of the OSI model, using SNMP and traceroute(8) respectively to gather the necessary data. This data was fed into the GraphViz graphing engine and topological maps of various parts of Rhodes' network were produced. Limitations were identified in the approaches taken, particularly in the use of SNMP to gather information at layer two.

The growth of Rhodes' network was tracked over a period of thirteen months, and the resulting growth rates of various subnets at the University was calculated. It was noted that the monitoring period was too short to provide a completely accurate picture of the development of the Universities network, particularly on those subnets that had experienced sudden spurts of growth. Two other uses were found for the information that was gathered by the monitoring application. The first of these was a means of providing an estimate of the availability or "uptime" of various systems at Rhodes. The second, and perhaps more significant, use was in the clearing out of stale DNS entries from the University's master DNS database.

The problem of finding specific hosts on a network was examined at two different levels, the physical location of a host within a building and the logical positioning of a host in relation to other hosts on the

network. The newly installed network in Hamilton building at Rhodes was used as an example of why these techniques are needed, and as a case study to test the proposed solution. The final result was a solution that used a web interface to provide a combined approach to both problems.

Many current network monitoring tools make use of simplistic, symptomatic methods of reporting faults they discover. The problems associated with these methods was discussed and an alternative was proposed. By using expert systems to perform some of the routine and tedious diagnosis tasks normally associated with tracing network faults, more accurate and useful fault reports can be generated. Particular attention was paid to keeping these reports concise. An approach to intelligently gathering data to produce these reports was also investigated. The wide variety of protocol implementations was presented as an obstacle to the traditional testing of services, and it was suggested that this may be an area where neural networks could be usefully employed.

Finally, all four of these applications were examined to provide an overview of how they interacted and depended on each other. Their position within the OSI network management model was examined to provide a clear indication of the roles that each application was intended to perform, and of how they related to existing network monitoring approaches. In addition, the problems associated with porting the developed applications to other networks was examined. Specific attention was paid to those applications that were not easy to port, and details were give of the possible snags and limitations that might be encountered.

9.2. Future Work

There are two obvious extensions to the work that has been covered in this project. The first is to improve the mapping of networks at layer two, and the second is to examine the possibility of using neural networks to test network services. Both these possibilities have been discussed to some extent in earlier chapters.

9.2.1. Separating infrastructure from hosts

One of the greatest problems with the network mapping approach taken by Section 4.2, and consequentially, the determination of logical location, as described in Section 6.1, is the need for a large amount of seed information. This data is required in order to allow the system to distinguish between network infrastructure (switches, routers, *et cetera*) and hosts on the network. It would be useful if the system could automatically make this distinction.

There are two possible methods for making this distinction: by examining the organisationally unique identifier, or by using the simple network management protocol.

The IEEE publish a list of organisationally unique identifers, and regulate their assignment and use. Several policies exist that both define and make recommendations about how these identifers should

be used. There is a possibility that network infrastructure can be separated from network interface controllers by examining the OUI. Research into the IEEE's assignment policies will determine whether this is possible.

It is more likely, however, that vendors group together various classes of network infrastructure, in much the same way that CIDR allows network administrators to aggregate network blocks. If some method could be established to determine these aggregations, and in particular which OUIs or aggregated MAC address blocks correspond to network infrastructure, the MAC address of a particular device could be used to make a decision on whether that device corresponds to infrastructure or a host.

The second approach that could be employed is the simple network management protocol. The SNMP MIB-II defines a `system.sysServices` variable which is intended to indicate the layer of the OSI reference model at which a device operates. In theory, this could be used to determine whether a device is network infrastructure or a host. It is not entirely accurate, however. Unix-like operating systems are often employed at layers three or four as routers and firewalls. Their SNMP agents will report them as operating at layer seven, however, since the operating system is capable of operating at this layer.

Some scope exists for investigating these SNMP agents and discovering whether useful information can be extracted from them in order to use them to determine whether a particular device should be considered network infrastructure or an end host.

9.2.2. Intelligent testing

Section 7.3.2 looks at the problem of testing that various network services are functioning correctly. It takes the straightforward approach of providing a set of test routines for common services, and using a simple TCP connect to test those services for which there are no existing test routines. This is often unreliable, since the ability to connect to a particular port gives no indication of whether the service running on that port is functioning correctly.

Ideally, a network testing system should understand all protocols and be able to test that they are functioning correctly using these protocols. This is not realistic, however. A more sensible approach would be to create a system that is capable of learning new protocols and using this learned information to test services.

The use of neural networks to achieve this was proposed in Section 7.3.2. It remains to be seen whether neural networks (or indeed, any other form of intelligent automata) could be successfully employed in this field, and this leaves scope for future work in this area.

9.2.3. Other work

The fields of network monitoring and network management are large, and a great deal of scope exists

for research and experimentation within these fields. This project chose and focused upon four specific aspects of the field of network monitoring, and examined them at a detailed level. There is room for expansion within these aspects, as the two previous sections outline. There is also, however, scope for work outside of the confines of what was done in this project.

Even within Rhodes, several other problematic areas of network management were identified. For example, the issues associated with maintaining up-to-date configurations on large numbers of layer two and layer three switch devices, particularly considering that in the University environment, these devices are sourced from a variety of vendors. An ideal solution would be a single management interface that automatically generates appropriate vendor-specific configuration files for the various devices located around campus.

The centralised, automatic configuration of devices is in itself a large area for research. Take the case of the University's firewall, for example. Students, staff, departments, and individuals have a vast variety of expectations and needs from the firewall. Those people who are less computer-literate expect it to protect their machines, and indeed it is required to, since often these people do not adequately maintain their machines. On the other hand, those people who are more knowledgeable about computers find the default firewall configurations restrictive, and often argue that it prevents them from being able to make use of certain facilities. As a result, exceptions are made for specific machines.

Problems arise when trying to manage these exceptions. Currently these exceptions are created by hand by one of the University's systems administrators. In addition, there are no facilities to expire the rules, so they often remain in place long after they are needed — this creates unnecessary security risks. It would be useful if this task could be automated to some extent. For example, users could be allowed to select from a number of typical options (such as running a personal web server) via a web interface. These user-selected options would have a default lifetime after which they would have to be renewed or they would expire and be removed. Any atypical requirements would still be referred to the administrators, but in a well configured system these would be few and far between.

These two examples present some idea of the scope that is available in these fields.

9.3. Conclusion

This work looked at Rhodes University's campus-wide local area network with a view to discovering typical network monitoring problems. The intention was to discover the limitations of current network monitoring and management solutions, and to develop innovative solutions to the problems that were identified. Together, these two areas formed the criteria by which suitable candidate problems were ascertained.

For each of the four areas that were investigated, a solution that fulfilled the prescribed criteria was realised. Shortfalls in some of the approaches taken were identified and discussed in the context of

providing a more complete solution to the problem. In addition, each target area was discussed in terms of the Open Systems Interconnect network management model in order that its network monitoring role be better understood.

Specific attention was paid to the issue of portability, and in particular, the problem of monitoring heterogeneous networks was investigated. The implications of such heterogeneity relates directly to the portability of the solutions developed in this project and as a result, care was taken to ensure that the impact of this problem would be minimised.

The outcome of this work was a set of novel tools and techniques that served four specific network monitoring needs better than the existing solutions to these problems.

Appendix A. SMS-based Reporting System

Appendix A reproduces a paper that was presented by the author at the Southern African Telecommunications, Networks and Applications Conference in September 2002. This paper describes a service that uses the simple object access protocol to provide a means for applications to send messages using the GSM short message service.

This SOAP service is used by the intelligent reporting system described in Section 7.3.3 to send notification of faults to the network administrator.

A bi-directional SOAP / SMS gateway service

Guy Antony Halse and George Wells, Rhodes University

Abstract

Many applications need the ability to do real-time notification when events occur. Often the people who need to be kept aware of events are in a remote location.

This paper looks at a bi-directional gateway between networked computers and the GSM short message service. The gateway is implemented as a web service, and uses the Simple Object Access Protocol to facilitate data communication.

The service interacts with a database in order to facilitate retrieval of sent or received messages, as well as provide accounting abilities.

It is intended as a practical proof-of-concept application demonstrating some of the capabilities of the Simple Object Access Protocol.

A.1. Introduction

Remote notification is a facility that many monitoring and control applications could benefit from. For example, network managers are often interested in being automatically informed of exception conditions that occur on their network. It was just such an application that was the birth of the service described in this paper

With more than eleven million cellular telephone subscribers in South Africa [1], it seems reasonable to assume that majority of professionals in the country have, or have access to, a cellular phone. With this in mind, the most obvious and effective way to provide notification is to use the cellular networks' short message service (SMS).

The problem, therefore, is to provide a way for computers to interface with the short message service. While discussing the issues involved in doing this, it became quickly apparent that requirements for remote notification such as this extended well beyond the authors' network monitoring application.

Rather than have many people investigating different approaches to the same problem, it was agreed that it would be better to provide a single, unified method of providing remote notification using the cellular network. The final approach adopted was a web service, which will now be discussed in detail.

A.2. SMS Overview

The Short Message Service (SMS) allows text-based messages to be sent to and from mobile telephones on a GSM network [2]. Each message has a maximum length of 160 characters. SMS messages are divided into two categories: Mobile Terminate (MT — where the SMS message originates from the network provider) and Mobile Originate (MO — where the consumer can send messages to other consumers) [3]. In the context of MO and MT messages, the consumer refers to the end-user, the person with a cell phone. This is as opposed to the network provider, who provides the consumer with such services. In this section, we are only concerned with MO SMS messages.

Typically, SMS messages are sent and received by cellular consumers using cellular telephone handsets. Cell phones are not the only devices that have this capability though. Anything that is capable of talking to a GSM network, in theory, has the ability to send and receive SMS messages. Since we are trying to interface a computer with the GSM network, it makes sense to use a device designed to do so, in other words a GSM modem.

Almost all GSM modems (and this includes many cell phone hand sets) use RS-232 as a transport protocol. On top of this, they use a protocol called the AT+ command set to communicate with their controlling devices. AT+ was defined by the European Telecommunication Standards Institute [4], and was designed to be a backward compatible set of extensions to the Hayes AT command set [5]. Device manufacturers are free to add their own extensions to this command set, and such extensions usually have an identifying prefix.

The communications hardware used to implement the service described in this paper was just such a GSM modem, specifically a modem based on the Wavecom (`http://www.wavecom.com/`) chipset. It was pleasing to find such modems manufactured by a South African based company called TeliMatrix (`http://www.matrix.co.za/products_gcom.htm`).

A.3. Simple Object Access Protocol

A.3.1. SOAP overview

The Simple Object Access Protocol (SOAP) is a simple XML protocol for exchanging structured information over the Internet. It was, and is still, actively being developed by several working groups of the World Wide Web Consortium (W3C). Their goal was to bring the concept of "web services" to their full potential [6].

The W3C provides a formal definition of a web service: "A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artefacts and supports direct interactions with other software applications using XML based messages via internet-based protocols." [7]

The World Wide Web is increasingly being used as a medium to transfer data between applications, and SOAP is designed to facilitate these transfers. It has been intentionally kept fairly simple, and as such does not contain complexities such as sequencing, etc. It has been likened to a "web datagram service" [8] insomuch as it provides encapsulation of standalone elements of information.

The data transferred by a SOAP datagram is represented in an XML document, which would typically have an associated schema defining its structure. The schemas defining SOAP itself are still being finalised, and the current state of this work is represented by a draft of SOAP version 1.2. Information on this draft is available from the W3C (`http://www.w3.org/TR/soap12-part0/`).

A.3.2. SOAP is not the only way

"There's more than one way to do it" is an oft-quoted phrase used to refer to Larry Wall's Perl. The same is true of the web services approach. Many different organisations and groups of people have their own ideas to achieve different parts of the web services concept. Some of these approaches, such as Microsoft's .NET framework are all-encompassing, where as others take a more focused, and thus simpler approach.

SOAP was used in this implementation for three main reasons: Firstly, SOAP is a project of the W3C, which is a widely respected authority on matters related to the web - they are, for example, the defining body for both HTML and XML. It seems appropriate, therefore, to choose a framework that is standardised by the same people responsible for defining the underlying mark-up language.

More significantly, SOAP is a definition, rather than an implementation. Developers are not tied to specific platforms or languages, which makes services based on it very flexible.

One of the philosophies behind SOAP is that it should be kept as simple as possible. It leaves dealing with the complexities of services to the services themselves. This helps avoid unnecessary overheads

in the data encapsulation. The lack of complications was the third reason for the decision to use the Simple Object Access Protocol.

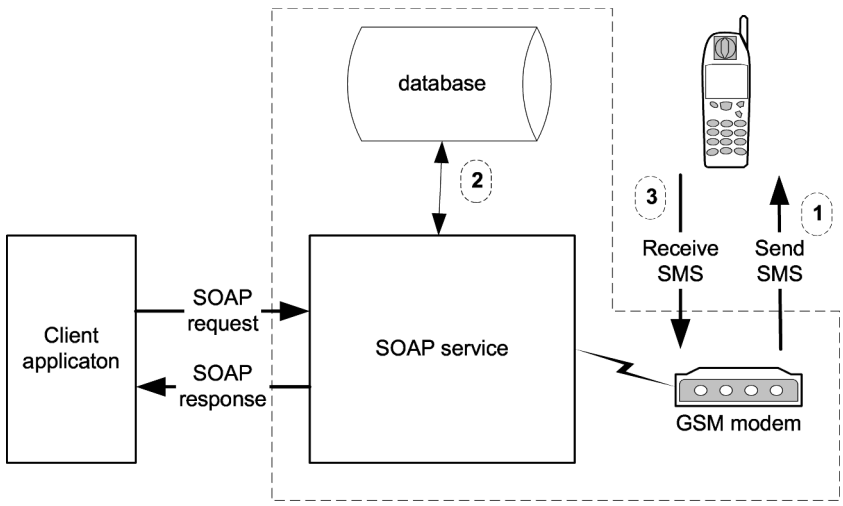
A.4. A Send Only Service

A.4.1. Background

The simplest form of notification is one-way communication — the intended recipient is provided with some information, but no method is provided for them to respond. In other words, the service can send the user information, but not the other way around.

This was the form of our “first pass” approach, and served to familiarise ourselves with the techniques required to communicate with the GSM modem. Care was taken, however, to ensure that the system was modular and extensible to cater for future growth. Figure A-1 below shows a block diagram of the complete service. This first implementation included all the links shown in the diagram except those labelled (2) and (3).

Figure A-1. Overview of the SOAP service



A.4.2. SOAP Encapsulation

In any web service, data needs to be transferred from the client to the service. Using SOAP as a transport encoding requires that this data be encapsulated as an XML document. The intention was to make this document as simple as possible, so as to allow the widest variety of applications.

The resulting schema allowed the client to send an XML document that encapsulated one or more SMS messages, each of which could have one or more intended recipients. A typical (and somewhat minimal) client request would be an XML document similar to the one in Figure A-2.

Figure A-2. Typical SOAP request to send SMS

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Body>
    <sms:sms xmlns:sms="...">
      <sms:sendsms>
        <sms:phone>0821234567</sms:phone>
        <sms:message>Hello World!</sms:message>
      </sms:sendsms>
    </sms:sms>
  </soap:Body>
</soap:Envelope>
```

This request shows a client asking for the message “Hello World!” to be sent to the owner of the cell phone whose number is 0821234567.

The web service would process this request, and would send the client a SOAP response indicating the status of this message. A typical SOAP response (and in fact, the response to the previous example) is shown in Figure A-3.

Figure A-3. Typical SOAP response

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Body>
    <sms:sms xmlns:sms="hellip;">
      <sms:sentsms>
        <sms:phone>0821234567</sms:phone>
        <sms:message>Hello World!</sms:message>
        <sms:status>ERROR</sms:status>
```

```
        <sms:id>S0015</sms:id>
      </sms:sentsms>
      <sms:newsms>R0006</sms:newsms>
    </sms:sms>
  </soap:Body>
</soap:Envelope>
```

The response contains the full information of the request, as well as various other fields containing information on the status of the message.

The most important of these is the status field. This contains information on whether or not the service was able to send the message. In the above example, this status is "`ERROR`", probably because 0821234567 in not a valid cell phone number. Success is indicated by an "`OK`" in this field. This information comes directly from the GSM modem.

A message identifier is sent back in the *id* field. This identifier is discussed in section on the database backend (Section A.6). The meaning of the *newsms* flag will become apparent in Section A.7 when we deal with received SMS messages.

A.4.3. HTTP transport

SOAP itself doesn't specify a transport method; this is left to the implementer. Just about any network-capable transport can be used. Internet file transfer protocols (such as FTP), mail protocols (such as SMTP and POP3), and online chat protocols (such as IRC and Jabber) have all successfully been used to transport SOAP datagrams. By far the most common method, however, is the hypertext transfer protocol (HTTP), which is the same protocol that is used to deliver web pages.

There are many SOAP over HTTP implementations, most of which include their own embedded web server. In the spirit of keeping it simple, it was decided to forgo these complications in favour of the existing web server software.

Thus, the service described in this paper runs as a common gateway interface (CGI) script on an Apache web server (`http://httpd.apache.org`). The CGI takes an HTTP "POST" method and passes the information (as well as meta-information about the client — server connection) to another program. This CGI service implementation simply reads information from the standard input stream and writes it to the standard output stream, and so could be written in just about any programming language. The Practical extraction and report language (Perl) was used to create the CGI interface in this service, mainly because of the author's familiarity with this language. Perl libraries for dealing with XML documents are widely available, and the XML::Parser module (based on the eXpat XML parser) [9] was used for this implementation.

A.4.4. Limitations

Like e-mail, the short message service employs a “best attempt” delivery method. This means that just sending an SMS message does not guarantee that intended recipient of that message will receive that message. Facilities exist to provide confirmation of delivery and notification of failure, in much the same way as e-mail servers generate status reports when mail cannot be delivered.

Unfortunately, this limitation means that the SOAP service cannot guarantee delivery of a client’s message. The status field that it returns to the client will indicate whether or not it has successfully managed to send the message, rather than whether the recipient has received it. No handling of delivery confirmation or failure notification currently exists in the service.

When the web service is communicating with the GSM modem, it does an exclusive lock on the serial port connecting the computer to the modem. This is necessary in order to ensure that the modem’s responses to commands that are sent are correctly interpreted.

This locking, however, means that only one client may be served at a time. Since the transaction with the web server takes a matter of seconds, this limitation shouldn’t pose much of a problem for low usage levels. However, if the service were to scale to high volumes of client connections properly, a queuing method would need to be employed to overcome this problem. This first pass implementation does not use such a queue, although the database described in Section A.6 could be used to achieve this.

A.5. Access Control and Authentication

The SOAP contains no facilities for providing security, accountability or billing, assuming rather that these are functions of the underlying transport layer. As mentioned in the previous section, SOAP datagrams can be transported over many protocols. Each of these transport methods has different methods of implementing access control and authentication.

Since we are using the HTTP protocol to implement the service discussed in this paper, we will focus exclusively on the security methods provided by this protocol, and specifically in the way the Apache web server implements these.

The most basic way of providing security for a web service is to limit the clients that can talk to it. This can be done based on the client machine’s hostname, domain name, IP address or network address. Host based access control, as this is commonly known, defines a set of trusted (and perhaps untrusted) hosts. If a host is trusted, anyone who uses that host will be able to use the service. Host based access control is a function of the web server, and its implementation may vary from server to server.

Obviously, this is not always ideal. Computers are often shared by many people, and sometimes one wants to be able to allow some of these users and not others. In the same way, one might want to allow a particular user to connect to the service from anywhere.

When an HTTP server wishes to restrict access to a particular web page or service, it sends the client a `WWW-Authenticate` header. If the client wishes to be granted access to the page, it must reply with an `Authorization` header containing a valid username and password for the service [10]. This authentication method is built into the HTTP protocol, so should be available on all web servers.

The main advantage of user-based access control is that it provides the service with a username that represents the client. This username can be used for many things, including message threading, accounting, etc. Some of these uses are discussed later in the paper.

One of the prime disadvantages of user-based access control is that the method requires that clear-text passwords be sent in the HTTP headers. This problem is not limited just to passwords, however. Using a standard HTTP connection, the content of the SOAP datagram is also prone to snooping. The simplest solution to this problem is to use transaction layer security (TLS), better known as the secure sockets layer (SSL).

TLS provides an encrypted medium to transfer the HTTP headers and data. This encrypted session is established before any information is transmitted, which solves both the problem of clear-text passwords, and the readability of the SOAP datagram. For this reason, the web service described in this paper is accessible via a standard secure HTTP (HTTPS) connection as well as normal HTTP.

A.6. A Database Backend

The most obvious extension to the simple service described in Section A.4 was to integrate it with a database. This database keeps track of which clients send what messages, and can later be used to handle incoming SMS messages as well. The connection labelled (2) in Figure A-1 above shows how this database links into the web service.

When a client sends an SMS message, a corresponding entry is created in the database. This entry contains the client's username, the date and time the message was sent, the hostname the client connected from, the status of the message, and optionally the message itself. A unique identifier represents each entry, and this identifier is returned to the client in the SOAP response.

The date and time, and the client's hostname effectively form a log of the activity, and are useful for tracing faulty connections or abuse of the service.

The *status* field is an indication of the status of the message. At present, this field simply confirms whether or not the message was sent out by the service. In future, however, it may serve to hold information on the delivery status of the SMS message.

Storing the message itself may be a contentious issue. The main advantage of this is that it allows a client to request messages that they have previously sent. This is done using the unique identifier that was returned to the client, and may be useful for providing context information. The disadvantage, obviously, is the possibility for invasion of privacy. The web-service's access control rules prevent users

from retrieving messages sent by other users. The safe storing of messages, however, depends on the service administrator being both trustworthy, and sufficiently securing the server from unauthorised attacks.

Having a database backend allows other applications to interface with the service. For example, an accounting application can use the database backend to work out SMS charges.

A.7. Dealing With Received SMS

A.7.1. Receiving SMS

There are two ways to handle incoming SMS messages, and each method has advantages and disadvantages.

The AT+ command set supports online notification of received SMS messages. In the same way as a Hayes AT compatible modem sends the client program a `RING` to indicate that there is an incoming phone call, an AT+ modem sends the client an `AT+CNMI` when a new SMS message is received.

The advantage of using this online notification is that the client becomes aware as soon as a new message is received. This eliminates any delay between the receipt of an SMS message and its being handled by the service. Unfortunately, such a system requires that either a daemon constantly poll the serial line for this notification, or we listen for a serial port interrupt. Either way, this daemon would have to handle both incoming and outgoing SMS messages. Such an approach requires that we implemented some method of inter-process communication between the serial daemon and the SOAP service, greatly increasing the complexity of the system as a whole.

The simpler approach is to periodically poll the modem for new SMS messages. This can be done using any system scheduling application such as cron(8) [11]. The disadvantage of this approach is that there may be a delay between the receipt of an SMS message and when it is processed.

In the case of the web-service that is the subject of this paper, the second approach was implemented. It was felt that the advantages brought by simplicity of implementation outweighed the disadvantage of a delay in the handling of new messages. This is because the SOAP does not provide for the service to initiate connections to a client. Thus the client itself must initiate any connection that results in notification.

Since we have no way of forcing a client to establish this connection, the known time delay between the receiving of a message by the modem and its processing within the service, is inconsequential compared to the delay between the service and the client.

A.7.2. Storing it in the database

The GSM modem is polled by cron(8) once a minute for new SMS messages. When a new message is received, it is stored in the database described in the previous section. As soon as it has been successfully stored in the database, it is removed from the GSM SIM card to free up room for new messages.

Incoming SMS messages require that the same sort of information be stored as is done for outgoing messages, with a couple of differences. In the incoming SMS table, it is essential that the full text of the SMS message is stored — otherwise there is no way for the client to ever retrieve it.

Rather than store information on whether a message was successfully received, the status field for incoming messages contains information on whether the stored SMS message has been read or not.

The username field contains information on the intended recipient of the message. This is worked out using a threading scheme, which is described below.

A.7.3. Threading

One of the most complex problems in receiving an SMS message is getting it back to the right client. This is compounded by the fact that all messages are sent and received from one cell phone number, meaning we have to somehow turn a many-to-one transport method into a many-to-many service. Thus, the basis for any threading or distribution method must be contained within the received messages themselves.

The mechanism this service uses to thread messages was based on an idea used in eXcell Technologies' SMS ↔ e-mail gateway, in which the user enters the desired e-mail address as the first word of an SMS message [12]. Both major networks in South Africa have subsequently used this idea to implement a similar service for their customers.

When a new message is received, the first word is extracted from the message and used as a tag. The first character of this tag is used to determine what sort of threading is to be done.

If an "`S`" is the first character of the tag, the service assumes that the rest of the tag contains the message identifier of a previously sent SMS message. It will then try and find the original sent message in the database, and if it is successful will route the newly received message to the sender of the original message.

In the same way, an "`R`" as the first character refers to the message identifier of a previously received message. This is useful, for example, to continue a previous message past the 160-character limit imposed by the SMS standard.

Sometimes the remote user may want to initiate a thread themselves. This is catered for by the "`!`" prefix. Any word prefixed with an exclamation mark is considered to be a service username, and messages are routed to that user.

It is possible that a message is received that has no valid tag on it. This could happen either when the user forgets to tag the message, or mistypes the tag. If the service cannot work out who "owns" the message, it marks it as a special broadcast type. Any authenticated user of the service can read broadcast messages.

A.7.4. Client notification

Once we know who the message is intended for, we face the next challenge; letting that client know they we have a message for them. As has been mentioned already, SOAP does not allow for server-initiated connections. For this reason, clients can only be notified of pending messages when they connect to the service.

The XML schema makes provision for this with a *newsms* flag. This flag can be set in any client response (an example of it is given in Figure A-3, and contains a list of message identifiers for unread messages. The client can then use these message identifiers to retrieve the waiting messages.

A.7.5. Client retrieval

Clients can retrieve any message from the database using a SOAP request containing the message identifier of the message they are after. An example of such a request is given in Figure A-4.

Figure A-4. Typical client request to retrieve SMS

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Body>
    <sms:sms xmlns:sms="]>...">
      <sms:readsms>R00120</sms:readsms>
    </sms:sms>
  </soap:Body>
</soap:Envelope>
```

The readsms field can contain a list of one or more SMS messages to retrieve. In addition, it can contain one of four pseudo-message identifiers; "`ALL`", "`UNREAD`", "`BCAST`" or "`SENT`". These identify groups of messages, and except for "`BCAST`", their meaning is fairly straightforward. The "`BCAST`" identifier retrieves any of the broadcast messages (messages without specified recipients) described in Section A.7.3.

A.8. Applications of the Service

The XML schema describing this web service was made available to postgraduates in the Department of Computer Science at Rhodes University, and as a result a number of applications for this service have come to light.

The most immediate application was to a co-author's own research, where this web service is used to notify network administrators of error conditions on the network. These administrators can then acknowledge receipt of the notification and post back status reports by replying to the SMS message they received.

One of the Masters students in the Department, a member of the IP telephony group at Rhodes, has used the web service described in this paper for the media portion of an example MGCP-complaint gateway, which allows H.323 clients (such as Microsoft's Netmeeting) to send SMS messages.

Many other applications have been proposed for a service such as this. For example, one could set up a client that was linked to a building's access control system. SMS messages could be used to control various access points, such as remotely unlocking a door.

In the same way, a typical computer intruder-detection system could be linked to the service. This could allow systems administrators to receive near real-time notifications of attempted security breaches of their machines.

A.9. Future Work

The SMS protocol does not guarantee delivery of SMS message, so at the time of sending a message there is no way of knowing whether the intended recipient will ever get that message. This has limited the status information provided by this service to indicate whether the message was successfully sent, rather than successfully received.

Like e-mail, however, the SMS protocol provides a way for the MO customer to request delivery confirmation. This confirmation is received in the form of an MT SMS message sent to the customer once the message has been successfully delivered.

The web service could be extended to request such confirmation, and then use the resulting MT SMS to update the *status* field of the SMS message stored in the database. This would allow a client of the service to check whether or not their SMS message has been successfully delivered.

A further extension to this would be to include a delivery notification flag, like the *newsms* flag, that gets sent to the client once delivery confirmation has been received.

Appendix A. References

[1] *Statistics of Cellular in South Africa (http://www.cellular.co.za/stats/statistics_south_africa.htm)*, Cellular Online, January 2002.

[2] *SMS (Short Message Service) (http://www.gsmworld.com/technology/sms/index.shtml)*, GSM Association, .

[3] *What is SMS? (http://www.gsmworld.com/technology/sms/intro.shtml)*, GSM Association, July 2000.

[4] *ETSI TS 127 007 - AT command set for User Equipment (UE) (http://pda.etsi.org/exchangefolder/ts_127007v050100p.pdf)*, ETSI, March 2002.

[5] *The AT Command Set Reference — History ()*, Frank Durda, 2002.

[6] *Web Services Activity (http://www.w3.org/2002/ws/)*, World Wide Web Consortium, January 2002.

[7] *Web Services Architecture Requirements (http://www.w3.org/TR/2002/WD-wsa-reqs-20020429)*, World Wide Web Consortium, April 2002.

[8] *Introduction to SOAP (http://www.perfectxml.com/Conf/Wrox/Files/frank_soap.pdf)*, Frank Manteck, .

[9] *The Expat XML parser (http://expat.sourceforge.net/)*, SourceForge, .

[10] *Webmaster in a Nutshell*, S. Spainhour and V. Quercia, 1996, 1st Edition, O'Reilly.

[11] *vixie-cron (http://sources.isc.org/utils/admin/cron_at/vixie-cron3.txt)*, Paul Vixie, January 1994.

[12] *service guide (http://www.excell.to/index.php?action=service_guide)*, eXcell Technologies, .

References

[BigBrother, 2002] *Big Brother: Installation and Configuration Manual (http://bb4.com/bb/help/bb-man.html)*, Quest Software, Inc., May 2002.

[Briscoe, 2000] *Understanding the OSI 7-layer model (http://www.itp-journals.com/nasample/t04124.pdf)*, John Briscoe, PC Network Advisor, July 2000.

[Cheswick, 1999] *Internet Mapping Project (http://www.cs.bell-labs.com/who/ches/map/)*, Bill Cheswick and Hal Burch, Bell Labs, 1999.

[Cisco, 2001] *Cisco Discovery Protocol MIB (http://www.cisco.com/univercd/cc/td/doc/product/fhubs/fh300mib/mibcdp.htm)*, Cisco Systems, August 2001.

[Cisco, 2002] *Network Management Basics (http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/nmbasics.htm)*, Cisco Systems, February 2002.

[CMU, 2000] *Simple Network Management Protocol: Software Technology Review (http://www.sei.cmu.edu/str/descriptions/snmp_body.html)*, Software Engineering Institute, Carnegie Mellon University, September 2000.

[CPR, 2001] *2001 Cost of Downtime (http://www.contingencyplanningresearch.com/2001%20Survey.pdf)*, Contigency Planning Research, Eagle Rock Alliance, Ltd, 2001.

[FOLDOC, 1993] *Free On-line Dictionary of Computing (http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?Free+On-line+Dictionary)*, Denis Howe, Imperial College Department of Computing, 1993–2002.

[Ford, 1987] *How machines think: A General Introduction to Artificial Intelligence*, Nigel Ford, Wiley, 1987.

[Fowler, 2000] *The AT&T AST OpenSource Software Collection (http://www.usenix.org/publications/library/proceedings/usenix2000/freenix/full_papers/fowler/fowler.pdf)*, Glenn S. Fowler, David G. Korn, Stephen S. North, and Kiem-Phong Vo, 2000 USENIX Annual Technical Conference, June 2000.

[FreeBSD, 2002] *The FreeBSD.org network (http://www.freebsd.org/internal/machines.html)*, The FreeBSD Project, October 2002.

[IEEE, 2002] *Frequently Asked Questions: Organizationally Unique Identifiers (OUIs) (http://standards.ieee.org/faqs/OUI.html)*, Angela Landron, Institute of Electrical and Electronics Engineers, June 2002.

[Irwin, 2001] *Bandwidth management and monitoring for IP network traffic: an investigation*, Barry V.W. Irwin, Rhodes University, April 2001.

[Levine, 1990] *AI and Expert Systems: A Comprehensive Guide*, Robert I. Levine, Diane E. Drang, and Barry Edelson, McGraw-Hill, 1990, Second Edition.

[MySQL, 2001] *MySQL Benchmarks (http://www.mysql.com/information/benchmarks.html)*, MySQL AB, June 2001.

[Nortel, 2002] *Optivity Portfolio (http://www.nortelnetworks.com/products/01/optivity/index.html)*, Nortel Networks, 2002.

[Oetiker, 1998] *MRTG: The Multi Router Traffic Grapher (http://www.usenix.org/publications/library/proceedings/lisa98/full_papers/oetiker/oetiker.pdf)*, Tobias Oetiker, Twelth Systems Administration Conference (LISA '98), December 1998.

[Patterson, 1990] *Introduction to Artificial Intelligence and Expert Systems*, Dan W. Patterson, Prentice-Hall International Editions, 1990.

[Paul, 1998] *CVS entry for src/sys/pci/if_rl.c (http://www.freebsd.org/cgi/cvsweb.cgi/src/sys/pci/if_rl.c?rev=1.4&content-type=text/x-cvsweb-markup)*, William Paul, The FreeBSD Project, November 1998.

[Pitt Turner, 2001] *Industry Standard Tier Classifications Define Site Infrastructure Performance (http://www.upsite.com/TUIpages/whitepapers/tuitiers.html)*, W. Pitt Turner, IV and Kenneth G. Brill, The Uptime Institute, December 2001.

[Rose, 1991] *The Simple Book: An introduction to management of TCP/IP-based internets*, Marshall T. Rose, Prentice-Hall, 1991.

[Shay, 1999] *Understanding data communications and networks*, William A. Shay, Second Edition, International Thompson Publishing, 1999.

[Tanenbaum, 1988] *Computer networks*, Andrew S. Tanenbaum, Second Edition, Prentice-Hall International, 1988.

[TechWeb, 2001] *Server 54, Where Are You? (http://www.techweb.com/wire/story/TWB20010409S0012)*, TechWeb News, CMP Media, April, 2001.

[Wellens, 1996] *The Simple Times (http://www.simple-times.org/pub/simple-times/issues/4-3.html): Towards Useful Management*, Chris Wellens and Karl Auerbach, SNMP Technology, Comment, and Events(sm), Vol 4, No 3, July 1996.

[W3C, 2000] *XML Schema (http://www.w3.org/XML/Schema)*, C.M. Sperberg-McQueen and Henry Thompson, World Wide Web Consortium, April 2000.

[W3C, 2003] *Extensible Markup Language (XML) (http://www.w3.org/XML/)*, Liam Quin, World Wide Web Consortium, January 2003.

Internet Standards Process

[BCP 9] *The Internet Standards Process*, S. Bradner, IETF, October 1996, Revision 3.

[BCP 12] *Internet Registry IP Allocation Guidelines*, K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel, IETF, November 1996.

[RFC 1067] *A Simple Network Management Protocol*, J. Case, M. Fedor, M. Schoffstall, and J. Davin, IETF, August 1988.

[RFC 1157] *Simple Network Management Protocol (SNMP)*, J.D. Case, M. Fedor, M.L. Schoffstall, and J. Davin, IETF, May 1990.

[RFC 1213] *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, K. McCloghrie and M.T. Rose, IETF, March 1991.

[RFC 1219] *On the Assignment of Subnet Numbers*, P.F. Tsuchiya, IETF, April 1991.

[RFC 1519] *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, V. Fuller, T. Li, J. Yu, and K. Varadhan, IETF, September 1993.

[RFC 2011] *SNMPv2 Management Information Base for Internet Protocol using SMIv2*, K. McCloghrie, IETF, November 1996.

[RFC 2060] *Internet Message Access Protocol: Version 4rev1*, M. Crispin, IETF, December 1996.

[RFC 2142] *Mailbox Names for Common Services, Roles and Functions*, D. Crocker, IETF, May 1997.

[RFC 2819] *Remote Network Monitoring Management Information Base*, S. Waldbusser, IETF, May 2000.

[RFC 3232] *Assigned Numbers*, Internet Assigned Numbers Authority, IETF, January 2002.

Glossary of Abbreviations

Unless otherwise indicated, definitions in this glossary are derived from the Free On-line Dictionary of Computing [FOLDOC, 1993].

ACL

Access Control List. A list of the services available on a server, each with a list of the hosts permitted to use the service.

ADSL

Asymmetric Digital Subscriber Line. A form of Digital Subscriber Line (DSL) in which the bandwidth available for downstream connection is significantly larger than for upstream.

ARIN

American Registry for Internet Numbers. One of a group of regional Internet registries responsible for the allocation of various Internet resources, such as IP addresses, and the formulation of consensus-based policies[1].

ARP

Address Resolution Protocol. A method for finding a host's Ethernet address from its Internet address. (RFC 826).

BCP

Best Common Practice. Those non-standards track RFCs which contain information on current practices and preferred approaches to solving common problems [BCP 9].

CDP

Cisco Discovery Protocol. A method used by Cisco routers and switches to perform automatic neighbour discovery [Cisco, 2001].

1. American Registry for Internet Numbers (`http://www.arin.net/`)

CIDR

Classless Inter-Domain Routing. A scheme which allocates blocks of Internet addresses in a way that allows summarisation into a smaller number of routing table entries. (RFC 1519).

CIR

Committed Information Rate. A guaranteed minimum throughput rate available on a frame relay or other network service. The network may start dropping packets when the data rate exceeds the CIR, but not before[2].

DHCP

Dynamic Host Configuration Protocol. A protocol that provides a means to dynamically allocate IP addresses to computers on a local area network. (RFC 2131).

DNS

Domain Name System. A general-purpose distributed, replicated, data query service chiefly used on Internet for translating hostnames into Internet addresses. (STD 2).

DSL

Digital Subscriber Line. A family of digital telecommunications protocols designed to allow high speed data communication over the existing copper telephone lines between end-users and telephone companies. Also known as xDSL.

DSLAM

Digital Subscriber Line Access Multiplexer. The generic term for the central office equipment where xDSL lines are terminated.

2. Encyclopedia of Networking and Telecommunications (`http://www.linktionary.com/c/cir.html`)

DTD

Document Type Definition. The definition of a document type in SGML or XML, consisting of a set of mark-up tags and their interpretation.

GSM

Global System for Mobile Communications. A standard for digital cellular communications (in the process of being) adopted by over 60 countries.

HTML

HyperText Markup Language. A hypertext document format used on the World-Wide Web. HTML is built on top of SGML.

HTTP

HyperText Transfer Protocol. The client-server TCP/IP protocol used on the World-Wide Web for the exchange of HTML documents. (RFC 2068).

IANA

Internet Assigned Numbers Authority. The central registry for various "assigned numbers": Internet Protocol parameters, such as port, protocol, and enterprise numbers; and options, codes, and types. (RFC 3232).

ICMP

Internet Control Message Protocol. An extension to the Internet protocol that allows for the generation of error messages, test packets, and informational messages related to IP. (RFC 792)

IEEE

Institute of Electrical and Electronics Engineers. A nonprofit, technical professional association based in the United States that develops, among other things, data communication standards[2].

2. Encyclopedia of Networking and Telecommunications (`http://www.linktionary.com/c/cir.html`)

IP

Internet Protocol. The network layer for the TCP/IP protocol suite widely used on Ethernet networks. (RFC 791).

IPC

Inter-Process Communication. Exchange of data between one process and another, either within the same computer or over a network.

IPSEC

IP Security. A protocol that provides security for transmission of sensitive information over unprotected networks such as the Internet.

ISDN

Integrated Services Digital Network. A set of communications standards allowing a single wire or optical fibre to carry voice, digital network services and video.

ISO

International Standards Organisation. A voluntary, non-treaty organisation founded in 1946, responsible for creating international standards in many areas, including computers and communications.

ITU

International Telecommunications Union. The ITU-T, the telecommunication standardisation sector of ITU, is responsible for making technical recommendations about telephone and data (including fax) communications systems.

LAN

Local Area Network. A data communications network which is geographically limited (typically to a 1 km radius) allowing easy interconnection of terminals, microprocessors and computers within adjacent buildings.

MAC

Media Access Control. The lower sublayer of the OSI data link layer. The interface between a node's Logical Link Control and the network's physical layer. A MAC address is the hardware address of a device connected to a shared network medium.

MIB

Management Information Base. A database of managed objects accessed by network management protocols.

MRTG

Multi-Router Traffic Grapher. Tobias Oetiker's network monitoring system [Oetiker, 1998].

NIC

Network Interface Controller/Card. An adapter circuit board installed in a computer to provide a physical connection to a network.

OID

Object IDentifier. Generally an implementation-specific integer or pointer that uniquely identifies an object.

OSI

Open Standards Interconnect. The umbrella name for a series of non-proprietary protocols and specifications, comprising, among others, the OSI Reference Model, Common Management Information Protocol and Services, and the Network Management Model.

OUI

Organisationally Unique Identifier. An OUI or "company id" is a 24 bit globally unique assigned number referenced by various standards. OUI is used in the family of 802 LAN standards, e.g. Ethernet, Token Ring, et cetera [IEEE, 2002].

RADSL

Rate Adaptive Digital Subscriber Line. A non-standard version of ADSL that allows modems to adapt the rate of transfer to match conditions on the line[3].

RFC

Request For Comments. One of a series, begun in 1969, of numbered Internet informational documents and standards widely followed by commercial software and freeware in the Internet and Unix communities.

RGB

Red, Green, Blue. The three colours of light which can be mixed to produce any other colour. Coloured images are often stored as a sequence of RGB triplets or as separate red, green and blue overlays.

RRD

Round Robin Database. The database used by Tobias Oetiker's MRTG [Oetiker, 1998].

SOAP

Simple Object Access Protocol. A minimal set of conventions for invoking code using XML over HTTP.

3. DSL Forum glossary (`http://www.dslforum.org/PressRoom/DSL_Glossary523.html`)

SEALS

South East Academic LibrarieS. A library service for the Eastern Cape Higher Educational Association (South Africa)[4].

SGML

Standard Generalised Markup Language. A generic markup language for representing documents. SGML is an International Standard that describes the relationship between a document's content and its structure.

SMS

Short Message Service. A message service offered by the GSM digital cellular telephone system.

SNMP

Simple Network Management Protocol. The Internet standard protocol developed to manage nodes on an IP network. (RFC 1157).

SQL

Structured Query Language. An industry-standard language for creating, updating and, querying relational database management systems.

TCP

Transmission Control Protocol. The most common transport layer protocol used on Ethernet and the Internet. (RFC 793).

telco

TELephone COmpany. A company providing telephone services to end users.

4. South East Academic Libraries (`http://www.seals.ac.za/`)

TTL

Time To Live. A field in the Internet Protocol header which indicates how many more hops a particular packet should be allowed to make before being discarded or returned.

UDP

User Datagram Protocol. An Internet standard network layer, transport layer and session layer protocol which provides simple but unreliable datagram services. (RFC 768).

URI

Universal Resource Identifier. The generic set of all names and addresses which are short strings which refer to objects (typically on the Internet). The most common kinds of URI are URLs and relative URLs.

URL

Uniform Resource Locator. A standard way of specifying the location of an object, typically a web page, on the Internet.

VLAN

Virtual Local Area Network. A logical grouping of two or more nodes which are not necessarily on the same physical network segment but which share the same network number.

W3C

World Wide Web Consortium. The main standards body for the World-Wide Web. W3C works with the global community to establish international standards for client and server protocols that enable on-line commerce and communications on the Internet. It also produces reference software.

xDSL

Digital Subscriber Line. See DSL.

XML

eXtensible Markup Language. An initiative from the World Wide Web Consortium defining an "extremely simple" dialect of SGML suitable for use on the World-Wide Web.

Colophon

This thesis was written in SGML using the DocBook markup language. Since this appears to be a fairly atypical approach, at least within Rhodes University, this colophon attempts to document the process.

DocBook

DocBook provides a system for writing structured documents, such as this thesis, using the standardised general markup language, SGML (or the extensible markup language, XML). It is particularly well suited to documents with a computer slant, but is not exclusively limited to them. The DocBook document type definition is maintained by the DocBook Technical Committee under the auspices of the Oasis consortium, and is freely available to all.

This document is written using the SGML version of DocBook 4.2.

One of the features of DocBook is that it separates presentation from content in any document. The DocBook DTD provides a way of marking up document *content*, and presentation is controlled by separate stylesheets. This allows DocBook documents to be published in many different formats, and for the presentation to be easily changed.

This document makes use of Norman Walsh's modular DSSSL stylesheets (version 1.73) to control *presentation*, which allow both screen and print versions of this document to be produced.

DocBook is well documented, both on the web and in print. *DocBook: The Definitive Guide* is written by Norman Walsh and Leonard Muellner and is published by O'Reilly & Associates. The SGML sources for this book, as well as HTML and PDF versions, are freely available via the world wide web. Extensive use was made of this book during the writing of this thesis.

Producing Different Document Formats

In order to produce documents of different formats, the DocBook SGML source has to be processed using a DSSSL (Document Style Semantics and Specification Language) engine. This engine renders the SGML content based on the presentation rules defined in the DSSSL stylesheet.

This document is processed using the OpenJade DSSSL engine.

The process of producing the different output formats is controlled using the FreeBSD documentation project's document build toolset. This is a set of Makefiles that automate the process of generating DocBook documents. The toolset is readily available from the FreeBSD web site.

The FreeBSD build toolset provides the ability to produce documents in the hypertext markup language (HTML), plain text format, PostScript (PS), Adobe's portable document format (PDF), TeX/LaTeX, and the device independent file format (DVI). The PostScript and PDF versions of the document make use of the JadeTeX macros. The HTML versions of the document make use of the World Wide Web consortium's HTML Tidy utility.

OpenJade, the DSSSL processing engine also allows for output in XML. This means that SGML DocBook documents can easily be converted to XML DocBook documents, allowing the use of XSL stylesheets as well as DSSSL ones. The FreeBSD documentation build toolset supports the use of XML documents with XSL stylesheets, but this approach was not used in the production of this document.

Some post-production of the TeX, PostScript and PDF version is done using the patch(1) utility. A patch to the TeX source was produced, and this was applied by the Makefile before the JadeTeX macros were used to produce other formats. Unfortunately, this patch was necessary in order to make the recto title page to conform to the University's formating regulations regarding the title pages of theses. A better way of achieving this would have been to correct the DSSSL stylesheet to produce the right output.

Fonts

The PostScript and Portable Document Format versions of this document use the Helvetica font for headings and the Times-Roman font for the body text. The base font size is 11 point. The HTML version makes use of the Verdana font if it is available.

Images

Including images in DocBook was perhaps the most complex part of creating this document. Fortunately, the FreeBSD documentation project has done a lot of work towards integrating images into their documentation build toolset Makefiles. In addition, Nik Clayton has written a fair amount of documentation on the process.

None of this documentation, however, covers the task of getting the original images into the right formats. There are two basic classes of images, vector-based and bitmap-based. Each of these types has its own intricacies involved in their conversion, and experimentation determined the best way to proceed with them.

For ease of reference, the images in this document were created in the following ways:

- Figure 3-1, Figure 7-1, Figure 7-2, and Figure A-1 were created with Microsoft's Visio. For each one of these, a PostScript image was created by using the Windows "print to file" option. PNG format images were generated using Visio's export options. Unfortunately, support for non-standard resolutions in Visio does not work entirely as expected, and for each image, the resolution had to be correctly set using JASC Software's Paint Shop Pro.
- Figure 3-3, Figure 5-1, and Figure 6-1 are screen-shots. These images were created as Windows bitmaps, and were edited in Paint Shop Pro. Once they were correct, they were saved as PNG images using Paint Shop Pro's PNG export filter. The Windows "print to file" option was again used to create PostScript versions, since Paint Shop Pro's PostScript export filter does not work correctly.
- Figure 4-1, Figure 4-4, Figure 8-1, and Figure 8-2 were generated using AT&T's GraphViz. GraphViz has the ability to generate both PostScript and PNG format images.

- Figure 5-2 was generated by Microsoft's Excel spreadsheet application. Excel offers the ability to export these charts to a web page, from which a PNG image can be extracted. The resolution of this image was corrected using Paint Shop Pro, and a PostScript image was created using the "print to file" option.

All PNG images were created at a resolution of about 300 dots per inch. These images are print-ready, and are suitable for use with PDF format output. PDFJadeTeX does not correctly support interlaced PNG images, so none of the images were interlaced.

In order to create PNG images more suitable for web pages, the ImageMagick convert(1) utility was used. The images are rescaled to a width of 600 pixels and were interlaced to give the appearance that they download faster.

The JadeTeX macros make use of Encapsulated PostScript images when generating PostScript output. The GNU version of GhostView was used to calculate the bounding box in each PostScript image, and to save these images as Encapsulated PostScript. These EPS images were tidied up using the eps2eps(1) utility.

Printing

In order to produce the final printed version of this document, the documentation build toolset was used to create a PostScript version of this document. This PostScript document was printed on a Hewlett-Packard HP4100TN laser printer. Those pages which contain colour images were printed on a Hewlett-Packard ink-jet printer.