

**NETWORK ACTIVITY AUDITING
USING LINUX RUNNING OFF
A LOOPBACK ROOT FILESYSTEM**

By

MOHAMED GHAUTH BIN MOHAMED HASSAN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

© Copyright 2008

by

Mohamed Ghauth bin Mohamed Hassan, 2008

CERTIFICATION OF APPROVAL

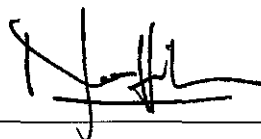
NETWORK ACTIVITY AUDITING USING LINUX RUNNING OFF A LOOPBACK ROOT FILESYSTEM

by

Mohamed Ghauth bin Mohamed Hassan

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:




Dr. Nor Hisham Hamid
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2008

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Mohamed Ghauth bin Mohamed Hassan

ABSTRACT

Areas where network traffic auditing helps network administrators is to identify sources of network activities with regards to the quality of data transmission (such as packet losses and latency) and quantity of data transmitted (such as absolute number of bytes and packets, as well as their rate of transmission per second), which are used to take the next step to remedy problems raised by them. In this project, we utilized a Debian GNU/Linux operating system running off a Loopback Root Filesystem as a network traffic auditing system. The project covers the design of the Linux system, use of Argus (Audit Record Generation and Utilization System—a network traffic auditing suite of tools), and the interpretation of the data gathered. The focus is to evaluate the designed system, analyze the data gathered and propose the next steps to improve the network traffic auditing system and the network it has audited.

ACKNOWLEDGEMENTS

I praise God for the opportunity to undertake this project. Many thanks to my supervisor, Dr Nor Hisham Hamid, for his extreme patience. I would also like to thank En Musa of the EE Dept for letting me use the lab facilities towards developing the system, and En Arfaisha and En Rahim of IT and Media Services for their time to meet me and explain some things regarding the UTP network. My gratitude goes to the EE FYP committee for their guidance, and my family members who have given the moral and material support throughout this project. Lastly, to all the people that make up the community in UTP that I have been friends with personally, or acquainted with virtually (through the UTP network), and met with even briefly, I appreciate every minor thing that have aided me towards doing this project, especially the people who have expressed a spectrum of views of the UTP network.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study.....	1
1.2 Problem Statement	2
1.3 Scope of Study	2
CHAPTER 2 LITERATURE REVIEW	3
2.1 Relationship between network traffic auditing and the TCP/IP infrastructure	3
2.2 How network traffic auditing affects the network neutrality debate	4
2.2.1 The two extremes of the network neutrality debate.....	5
2.2.2 Moderate approaches to the network neutrality debate	5
2.3 Usage of network traffic auditing at several universities.....	6
2.3.1 The root of the bandwidth congestion problem.....	7
2.3.2 Decisions made to regulate user traffic	7
2.4 Several Linux issues to consider	8
2.4.1 General information about Linux and Debian GNU/Linux.....	8
2.4.2 Background and some concerns of the Linux Loopback Root Filesystem	10
2.5 Argus on Linux as a network traffic auditing platform.....	11
2.5.1 Argus as a network traffic auditing tool	11
2.6 UTP administration's take on network traffic controls.....	12
2.6.1 IT and Media Services.....	12
2.6.2 Response from the Rector.....	13
CHAPTER 3 METHODOLOGY	14
3.1 Preparation of the Linux Loopback Root Filesystem.....	14
3.1.1 Linux Loopback Root Filesystems versions.....	14
3.1.2 Initial setup to get a fundamental Linux Loopback Root Filesystem booted	15
3.1.3 Further setup to ensure that the Linux Loopback Root Filesystem will shutdown/reboot safely	17

3.1.4 Additional customizations to improve the Linux Loopback Root Filesystem	18
3.2 Physical setup of a network traffic auditing system.....	18
3.2.1 Installation of Linux and Argus.....	19
3.2.2 Hub connection of Argus to the UTP network	20
3.3 Observing traffic with Argus.....	20
3.4 Parsing Argus generated data with ra* programs.....	21
3.5 An approach to internet bandwidth monitoring	21
3.6 Usage of Argus to monitor the traffic at Village 2 Block E.....	21
CHAPTER 4 RESULTS AND DISCUSSION	23
4.1 Internet bandwidth monitoring results	24
4.2 Results of the Village 2 Block E normal traffic observations with filters applied.....	26
4.3 Drawbacks of the Linux Loopback Root Filesystem as a network traffic auditing system	28
4.4 How to address the drawbacks of the system.....	28
4.5 Other benefits of the system.....	29
4.6 Some thoughts regarding the progression of the project.....	30
4.7 Outlook on future studies of network traffic auditing.....	30
4.7.1 Change in the management's view of the UTP network	31
4.7.2 Making the IT administration of the network more robust....	31
CHAPTER 5 CONCLUSION AND RECOMMENDATIONS	33
REFERENCES.....	34
APPENDICES	36
Appendix A Customized initramfs.....	37
Appendix B Old Version Scripts.....	51
Appendix C New Version Scripts	57
Appendix D Scripts used to generate Argus Graphs.....	61

LIST OF TABLES

Table 1 Comparison between OSI seven-layer model to the TCP/IP model.....	4
Table 2 Details of Argus network traffic capture activities	23
Table 3 Ten most popular traffic according to ramon.....	23
Table 4 Port 80 traffic contribution sorted by most generated to least generated for Wednesday April 4 th 2007	24
Table 5 Port 80 traffic contribution sorted by most generated to least generated for Thursday April 5 th 2007.....	24

LIST OF FIGURES

Figure 1 Multiple computers connected to a hub, which connects to the UTP network	20
Figure 2 Several graphs generated by the scripts in Appendix D.....	25
Figure 3 Graphs showing rate of bits received on the V2E network port that do not belong to the port, between 11 am and 5 pm for May 5 – 8, 2008.....	26
Figure 4 Graphs showing rate of packets received on the V2E network port that do not belong to the port, between 11 am and 5 pm for May 5 – 8, 2008.....	27

LIST OF ABBREVIATIONS

Argus	Audit Record Generation and Utilization System
IP	Internet Protocol
MAC	Media Access Control
P2P	Peer-to-Peer
PDU	Protocol Data Unit
QOS	Quality of Service
TCP	Transport Control Protocol
UDP	User Datagram Protocol

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Network traffic auditing is the process of examining data packets to gain understanding of what goes on with the network and later used to decide on a proper course of action. A basic form of network traffic auditing is done by web proxy servers which censor based on keywords used to access a website. A more process intensive form of network traffic auditing is deep packet inspection, whereby a number of packets are monitored and its payload analyzed by the observing party to detect harmful traffic, such as analyzing specific methods used in a hacking attempt or an internet worm propagation while they are in progress.

Network flow analysis is another form of network traffic auditing. Specific attributes of the flows, packets transmitted on the network contributing to conversations between hosts, are recorded, without looking at the payload. This helps to identify generators of network traffic and the quality of the flows. For example, by only looking at the number of bytes sent and received over a period of time, hosts which are generating the most amount of traffic on the network can be identified.

Argus (Audit Record Generation and Utilization System) is an open source network flow analyzer which runs on many UNIX-like systems. Coupled with the Debian GNU/Linux operating system, we would be able to create a network traffic auditing system at a low monetary cost. Considering that most people in UTP may not be familiar with Linux, a system that makes deploying Debian GNU/Linux easy should be considered. One area which intimidates new users is repartitioning a hard drive to install Linux. Loopback Root Filesystem, a method to run Linux without repartitioning has been done before, and it is redesigned for use in this project.

1.2 Problem Statement

This project addresses the following questions:

1. How can we run Debian GNU/Linux on a Loopback Root Filesystem?
2. How beneficial is the system and what drawbacks exist with the implementation of this system? How can we address these issues?
3. How useful is Argus, while running on the Linux system, at monitoring the network activities of computers connected to a network segment?
4. What can we learn about the network from the data we gathered with the Argus server and analyzed using the Argus client tools?

1.3 Scope of Study

In this project, we done the following:

1. Design and setup a Debian GNU/Linux system to run on a Loopback Root Filesystem.
2. Setup Argus to observe network traffic, which include:
 - a. setting up Linux and Argus on different computers.
 - b. test the setup at the Data Communications and Networking Lab (23-02-13), observing traffic flow of four computers connected to a hub.
 - c. test the setup on one computer connected to the residential network of Village 2 Block E.
 - d. use Argus to monitor traffic flowing towards the computers, which are stored in Argus data files.
3. Run Argus client tools to parse the Argus data files. Interpret the output.
4. Identify the benefits and drawbacks of the Linux system.

In the next chapter, we will discuss some issues for the basis of our project and explain some of the tools used in this project. We discuss the methodology used to achieve the objectives of the project in Chapter 3 and discuss the results in Chapter 4. We conclude our findings in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

In discussing network traffic auditing and running Argus on Debian GNU/Linux on a Loopback Root Filesystem, we have come across several issues, namely:

1. Relationship between network traffic auditing and the TCP/IP infrastructure,
2. How network traffic auditing affects the network neutrality debate,
3. Usage of network traffic auditing at several universities,
4. Several Linux issues to consider,
5. Argus on Linux as a network traffic auditing platform, and
6. UTP administration's take on network traffic controls.

2.1 Relationship between network traffic auditing and the TCP/IP infrastructure

Before we begin to discuss network traffic auditing, we have to understand where it falls in the TCP/IP model. Table 1 is adapted from a comparison made between the Open Systems Interconnection (OSI) seven-layer model and the TCP/IP model. [1]

The table provides some examples to relate layers within the TCP/IP model to the corresponding OSI layers. Typically, network traffic auditing software has access to Media Access Control (MAC) Protocol Data Units (PDUs), thus it works at the Layer 2 (Data link layer) of the OSI model. In the case of network flow analysis software such as Argus, the software is able to examine the MAC and Internet Protocol (IP) source and destination addresses, as well as identifying the Transport Control Protocol (TCP) and User Datagram Protocol (UDP) packets, thus it audits layers up to layer 5 (Sessions layer). [2] With deep packet inspection done by software such as Wireshark, information at layer 7 (Application layer) can be examined. [3]

Table 1 Comparison between OSI seven-layer model to the TCP/IP model

OSI	TCP/IP
Application	<p>Application</p> <ul style="list-style-type: none"> - usage of protocols such as HTTP, HTTP over SSL and FTP belong to the application layer. - another example is an implementation of a multiplayer game communication protocols. - the presentation layer part is identified by its portability across different systems. - the application contains the session layer when reliability of transmitting UDP/TCP packets is delegated to it.
Presentation	
Session	
Transport	<p>Transport (host-to-host)</p> <ul style="list-style-type: none"> - usage of transport protocols UDP and TCP. - TCP also belongs to the session layer as it is used to ensure reliable transmission of data.
Network	<p>Internet (IPv4)</p>
Data Link	<p>Network Access</p> <ul style="list-style-type: none"> - partially belongs to the network layer when we consider usage of routers/switches that utilizes virtual LANs. - data link parts provided by LLC and MAC.
Physical	<p>Physical</p> <ul style="list-style-type: none"> - e.g., ethernet cable, bit-by-bit signal encoding such as Manchester encoding, carrier sense and collision detection.

2.2 How network traffic auditing affects the network neutrality debate

The network neutrality debate revolves around the idea of whether or not a network service provider should be allowed to take discriminatory action on any network traffic that flows through its part of the network. In turn, the issue leads to the question of whether government regulations should be used to enforce network neutrality to avoid discriminatory practices taken by network service providers.

The act of discrimination itself is defined differently depending on context. In [4], discrimination is defined as when network traffic or network users are treated differently from others. While in [5], discrimination is defined to occur when differences in costs of using a network service are inconsistent with the costs of providing the network service.

2.2.1 The two extremes of the network neutrality debate

Opinions on the mostly U.S.-centric debate falls on a wide range of spectrum. On one end, proponents of government intervention to force network service providers to not discriminate, such as the Save the Internet coalition [6], include individuals and several civil society groups. They are of the opinion that without government intervention, network service providers will discriminate network traffic which is not in the best interest of the public, as it will hinder free speech and innovation, slowing the advancement of internet enterprises. They cite Google as an example of an innovative internet company which flourished while the internet is network neutral.

At the other end of the spectrum, opponents of any government regulation include several network service providers, network equipment makers, manufacturers and trade associations, under the banner of Hands off the Internet [7]. This coalition claims that government regulations are not necessary, and letting the government regulate the network would be opening the floodgate of more government regulation of the internet, which had flourished without any government's intervention. To the coalition members, discriminatory practices employed by network service providers are beneficial to all parties involved and falls within the right of a network service provider to manage limited resources (network bandwidth) that is often in high demand, especially when handling the bandwidth needs of peer-to-peer (P2P) application users. This is where network traffic auditing plays its part.

2.2.2 Moderate approaches to the network neutrality debate

While [6] and [7] represent the extremes of the network neutrality debate, the discussions in [4] and [5] evaluate the merits and drawbacks of both extremes. These discussions brought to light several issues suppressed or disregarded in the debate of network neutrality.

The following are some issues discussed in both papers:

1. Not all discriminatory actions are harmful to the users. For example, the ability to detect and isolate traffic that is causing more congestion when a segment of a network is being heavily used is very useful to alleviate or solve such a problem. It is not necessary to ban such traffic from the network, but it

is much easier to stop such traffic when this problem occurs.

2. In light of discriminatory measures employed by network service providers, users may have the option to circumvent such measures. The effectiveness of the methods used to circumvent such measures would depend on how users are able to pull together resources, such as connectivity to other users from alternative service providers through an alternative wireless network connection.
3. The nature of network traffic discrimination permits the service provider to take action that is beneficial to all users—for example, to mitigate the spreading of a prevalent computer worm on the network, as well as detrimental to some or all users of the network—such as putting traffic from users who do not sign up for a service plan in a lower priority even though the service provider is able to provide the bandwidth to serve the user.
4. Although legislation can be used to force network service providers not to discriminate, enforcement of such a law comes after the act of discrimination has been done and determining if such an action is proper or not may be a subjective decision.

Thus, the outcome of the network neutrality debate still cannot provide a definitive answer by endorsing legislation that would penalize all acts of network discrimination or allowing discriminatory acts that would harm the interests of the public.

2.3 Usage of network traffic auditing at several universities

Looking at the situation of the need for some network traffic control, many universities faced the same situation. Here we will discuss the methods used at the following universities:

- Carnegie Mellon University (CMU), Pennsylvania, USA [8]
- Brandeis University (Brandeis), New York, USA [9]
- University of Waterloo (UW), Ontario, Canada [10]

The following subsections illustrate the reasons why similarities exist between the approaches taken at these universities.

2.3.1 The root of the bandwidth congestion problem

All the universities above stated that their bandwidth congestion problem was caused by the use of P2P programs to transfer files between P2P network users. It was stated that only a minority of students are users of the P2P network, but the ability of these programs to generate a great amount of traffic causes all the bandwidth resource of a university to be completely used up.

P2P programs initiate as many connections as possible and will drop a connection as quickly as possible, both of which overwhelm the routers. The act of dropping a connection is registered as a packet loss by the routers. As routers are configured to retransmit packets upon packet loss, combined with the high volume of connections initiated and dropped, the number of retransmits causes latency and greatly decreases the performance of the network.

2.3.2 Decisions made to regulate user traffic

Ideally, the universities would have preferred that the users would regulate their use of the network. In reality, steps have to be taken by the network administrators to return order to the network. These cover essentially three methods:

- Different bandwidth allocation between academic and residential networks
 - o This ensures that congestion in the residential network does not affect the performance of the academic network.
 - o Although it helps users in the academic network, users of the residential network still suffer performance degradation.
- Separate traffic according to the Quality of Service (QOS) needed
 - o Time critical network traffic is given higher priority compared to identifiable P2P or unidentifiable network traffic.
 - o Traffic shaping or allocating different kinds of traffic different bandwidth, such as a tiny fraction of internet bandwidth for all P2P traffic.
 - o QOS provided from the features available on Cisco routing hardware used on these networks [11].
 - o Bigger networks, such as at CMU, employ NetEnforcer, hardware dedicated specifically for traffic shaping.

- Audit all traffic generated by a user
 - o Some Cisco hardware provide some means of measuring how much traffic is generated by hosts connected to its switches.
 - o The network traffic auditing tool Argus has been used at CMU to measure users' bandwidth usage.
 - o By using the information gathered, network administrators can take appropriate action to inform users of any use or misuse, warn users of non-compliance or decide on terminating user privileges of the network.

Of the measures used above, designing a network traffic auditing system using Argus seemed to be a viable tool to develop for a student project. In the next few sections we look into how we can use Linux and Argus to design such a system.

2.4 Several Linux issues to consider

Although Linux covers a wide variety of subjects, we will discuss only a few issues important to our research. This covers the general information about Linux and Debian GNU/Linux, and the background and some concerns of the Linux Loopback Root Filesystem.

2.4.1 General information about Linux and Debian GNU/Linux

Before using Linux, users would need to choose a Linux distribution to run. This choice depends on the tasks the user needs to do. This flexibility makes it suitable for different types of operating systems applications, such as:

- a fresh installation of a Linux distribution on an empty formatted partition, commonly done to install Debian GNU/Linux [12].
- a bootable rescue CD, such as PLD Linux Rescue CD [13].
- a Linux system for first users of Linux, which runs from CD-ROM and the computer's RAM, such as Knoppix [14].
- a Loopback Root Filesystem, a complete Linux distribution installed on a single big file on a hard disk partition used by another operating system. The file can be booted to run Linux as if it resides on its own partition [15], [16].

For this project, we have chosen Debian GNU/Linux as the platform to use. Aside from being a Linux-based operating system able to run Argus, Debian supports a wide variety of software packages. Fundamentally, the Debian GNU/Linux operating system goes through these stages or distribution:

- Debian unstable:
 - o new and cutting edge software are first packaged and tested here
 - o not all packages said to be available on Debian are available here
 - o bugs are expected, and updates are made often
 - o not suitable for production standard installations
- Debian testing:
 - o came from Debian unstable software passing some criteria for widespread testing
 - o not all packages said to be available on Debian are available here
 - o bugs are not unexpected, but updates might not be as often as Debian unstable
 - o might not be suitable for production standard installations
- Debian stable:
 - o what becomes of Debian testing after a “freezing” period has passed
 - o almost all packages expected to be available exist in this distribution
 - o updates are rare, but option is available for users to update packages patched for security reasons
 - o primary choice for production standard installations
- Debian old-stable:
 - o what becomes of Debian stable after a new “freezing” period has passed of the current Debian testing.
 - o packages available might not be as up-to-date as Debian stable
 - o security patches and updates are available within a year after the new Debian stable is released
 - o installations of this distribution are considered for upgrade

Which Debian distribution chosen is not decided by getting the newest or most currently stable version. For example, if a legacy application is not available on the stable version, an installation of an old-stable version might be sufficient. Same applies to a new application that might not be available on the stable version, of

which the testing version would be a better candidate for installation.

Also, the installation of Debian can also be done using Debootstrap, a software installation system without using a boot disk. Debootstrap reduces the time required to get a Debian system up and running.

For this project, it was first decided that our system will run on the Debian testing distribution, but our system was not updated often. As that distribution has been frozen and has since become a stable version, it was decided that the system should be made to run on a new installation of the new stable distribution.

2.4.2 Background and some concerns of the Linux Loopback Root Filesystem

A typical Linux Root Filesystem exists as a standalone Linux installation inside an ext2/ext3 formatted hard disk partition, similar to ordinary Windows installations—one Windows operating system per FAT32/NTFS formatted hard disk partition.

However, a Linux Loopback Root Filesystem is a special case of Linux Root Filesystem. It does not exist as an installation on an ext2/ext3 partition, but exists as a Root Filesystem image file on a hard disk partition which it has read and write access.

Guides to install Linux Loopback Root Filesystem can be read from [15] and [16], but the former is very much outdated (in 1999 or 2000), while the latter covers installation on a FAT32 partition.

The Linux NTFS Project [17] has provided limited write support for the NTFS partition in the Linux kernel. But its cautious outlook which forces read-only access to an NTFS partition previously unmounted cleanly (i.e., after a power failure) would disable the use of a Linux Loopback Root Filesystem on the partition.

Thus our system should be able to address these concerns during its operation from startup to shutdown.

2.5 Argus on Linux as a network traffic auditing platform

Audit Record Generation and Utilization System (Argus), is a network traffic auditing tool used to track and report the behavior of network traffic flowing through a network it observes [18]. It runs on several platforms, and Linux is one of them. In this section, we will discuss why Argus was chosen for this project.

2.5.1 Argus as a network traffic auditing tool

Argus was chosen for this project for its ability to monitor bandwidth utilization of hosts on a network, aside from being an Open Source software. It can also serve as a network forensics tool, being able to keep historical information about any traffic recorded on a network segment.

Argus stores statistics derived from information within “flows”—groups of inter-related packets. From a flow, the following information is recorded:

- start and end times
- protocol
- source and destination addresses
- source and destination ports
- direction of flow initiation (either one side exclusively initiate a connection, or both sides actively query each other, if possible to detect)
- number of packets and total number of bytes moved in each direction
- which state a flow is in when it is recorded. The following applies to all protocols except ICMP:
 - o request/initial state
 - o accepted state
 - o established/connected state
 - o closed state
 - o timeout state

While observing traffic, Argus can also be instructed to generate statistics regarding the measures of flow:

- loss rate of packets / number of retransmits
- delay between packets

- jitter, the variation of the delay between packet arrival times [19].

For our analysis, we are interested in constructing the Linux Loopback Root Filesystem and exploring some of the capabilities of Argus above. We explore this matter further in the next chapter.

2.6 UTP administration's take on network traffic controls

With the situation of the UTP network always being questioned by the students, we were able to get feedback from both UTP IT and Media Services executives [20] and the Rector of UTP [21].

2.6.1 *IT and Media Services*

From the meeting with the executives, the following can be said of the situation of the network:

1. The cause of users experiencing great latency while browsing pages off the internet is not only caused by some users using more of the internet connection, but also caused by the heavy traffic generated by activities between computers within the internal UTP network. This is the result of the design of the UTP network itself, which routes all traffic between different residential villages (i.e., Village 1 and Village 4) along the same path as internet traffic from the respective blocks. As the traffic between the residential blocks are higher in volume, caused by students participating in file sharing activities (such as using DirectConnect/DC++ protocols), compared to internet traffic, congestion occurs at the switches which handle the packet traffic queue.
2. Although bandwidth usage monitoring would detect users who are using more of the internet bandwidth, it would introduce the situation of underutilization—whereas all users who need to use the resource cuts back on their use such that they are using within their limits, the bandwidth available is more than enough to cater for all users at that particular point in time, which is also a waste of resource.
3. IT and Media Services is capable of monitoring bandwidth utilization using protocol analyzers such as Fluke, but it is still looking for any analysis tool

that can provide packet transmission efficiency information.

4. Although UTP may emulate steps to handle internet congestion taken by some universities outside of Malaysia, particularly those in the U.S. or Canada, it may not be suitable to apply their solutions to our problems, considering that the nature of their networks and ours are very much different.
5. The reason proxies used at the residential villages are very limiting is to get the students to access the internet from the many laboratories on campus, as the students may opt to use the Novell BorderManager proxies which do not impose as many restrictions as the proxies used at their dorms.

IT and Media Services are aware that the UTP network is facing problems. Although prior to this report, we may question why they seem to not take any action, we do now understand why they decide not to take action.

2.6.2 Response from the Rector

The Rector has highlighted several issues regarding activities on the network. The feedback from him echoes the sentiment expressed by IT and Media Services above, highlighting achievements regarding how the administration was able to handle network problems faced by UTP several years ago and has improved the network situation since then. The current problem now is the usage of Internet Relay Chat (IRC) and DirectConnect/DC++, among other things, causing network congestion.

In the next chapter, we explore the methodology used for this project in the light of the issues highlighted above.

CHAPTER 3

METHODOLOGY

In this chapter, we explain the steps taken to build a Linux Loopback Root Filesystem to use Argus. There are two primary steps:

1. Preparation of the Linux Loopback Root Filesystem
2. Setup of Argus and usage of the Linux Loopback Root Filesystem as a network traffic auditing system

3.1 Preparation of the Linux Loopback Root Filesystem

Before we are able to use a Linux Loopback Root Filesystem, we need to first design and build this system from zero.

3.1.1 *Linux Loopback Root Filesystems versions*

Towards completing this project, two different versions of Linux Loopback Root Filesystems were created:

- Old version
 - o Based on Debian testing (codenamed Etch) as of May 2006.
 - o Last updated in January 2007.
 - o Created on FAT32 partition with a file size of 2047MB.
 - o Setup using ‘debootstrap’ while running Knoppix as the host operating system.
 - o Shutdown/Restart follow similar procedures as normal Debian installation—able to use init 0 or init 6 or ctrl-alt-delete from tty.
- New version
 - o Based on Debian stable (codenamed Etch) as of August 2007.
 - o Last updated in January 2008.
 - o Created on FAT32 partition with a file size of 4095MB.
 - o Setup using ‘debootstrap’ while running ‘old version’ above as the

host operating system.

- Shutdown/Restart does not follow procedures as normal Debian installation—unable to use init 0/init 6/ctrl-alt-delete from tty.
- root has to type 'init 7' from the command line to initiate Shutdown/Reboot.

The reasons to create two different versions:

- the old version being very outdated, and
- recreating the system from zero is less cumbersome than upgrading, while avoiding upgrading mishaps.

The reason the shutdown/reboot scheme has changed is caused by the behavior exhibited by the new version when the old scheme was used. A working solution for the new version requires us to disable init 0/init 6/ctrl-alt-delete from tty.

3.1.2 Initial setup to get a fundamental Linux Loopback Root Filesystem booted

The following steps are required to create a Linux Loopback Root Filesystem. These actions are done while running Knoppix (to prepare the old version) and the old version of Debian (to prepare the new version), both of which need to be connected to the internet to download files:

1. Mount the host FAT32 filesystem:

```
mount -t vfat /dev/hda5 /mnt
```

2. Use 'dd' to create a file filled with zeroes the size we want (2047MB or 4095MB):

```
dd if=/dev/zero of=/mnt/debianS0.img bs=1M count=4095
```

3. Create an ext3 filesystem on the file:

```
mkfs.ext3 /mnt/debianS0.img
```

4. Mount the filesystem to access the file:

```
mount -t ext3 /mnt/debianS0.img /debilo -o loop
```

5. Populate the filesystem using debootstrap

```
debootstrap etch /debilo http://ftp.au.debian.org/debian
```

6. After debootstrap is done, change the root into the /debilo directory

```
chroot /debilo /bin/sh <everything else after this is done in /debilo>
```

7. Edit the /etc/apt/sources.list file using nano, and follow the instructions

```
nano /etc/apt/sources.list <press enter and add the following lines>  
deb http://ftp.au.debian.org/debian etch main contrib non-free
```

```
deb http://security.debian.org etch/updates main contrib non-free
<save the file by pressing Ctrl-O and answering Y, exit with Ctrl-X>
```

8. Run the following command to update and install a kernel image:

```
aptitude update && aptitude install kernel-image-2.6-686
```

9. Customize the initramfs to be able to boot the Linux Loopback Root Filesystem.

- a. Customizations made are listed in Appendix A
- b. The difference between the old version and new version with regards to the initramfs lies with the changes made in the scripts/local-bottom/debilo file. In Appendix A, the new version is named as scripts/local-bottom/new-debilo, and a context difference is shown as local-bottom_debilo.diff.
- c. The initramfs is generated as follows:

```
update-initramfs -c all
```

10. Further modifications are needed before the initramfs can be used:

- a. Extract the initramfs cpiogz archive (assuming its name is /boot/initrd.img-2.6.18-4-686) into /tmp/initram:

```
mkdir /tmp/initram && cd /tmp/initram
<by now, all the following activities
are done inside the /tmp/initram directory>
gzip -dc /boot/initrd.img-2.6.18-4-686 | \
cpio -i -d -H newc --no-absolute-filenames
```

- b. Edit the file 'init' as follows:

```
nano init
<add the following line inside the file:>
[ -f /scripts/initline ] && . /scripts/initline || \
<between the following lines>
maybe_break init
exec run-init ${rootmnt} ${init} "$@" <${rootmnt}/dev/console
>${rootmnt}/dev/console
<resulting with the following>
maybe_break init
[ -f /scripts/initline ] && . /scripts/initline || \
exec run-init ${rootmnt} ${init} "$@" <${rootmnt}/dev/console
>${rootmnt}/dev/console
<and save the file>
```

- c. Regenerate the cpiogz archive:

```
find . | cpio --quiet --dereference -o \
-H newc | gzip -9 > /boot/initrd.img-2.6.18-4-686-new
```

11. Prepare bootable media to boot the Linux Loopback Root Filesystem using `syslinux`

- a. Install a `syslinux` bootsector on a FAT formatted USB drive (assuming here it is named `/dev/sda1`):

```
syslinux /dev/sda1
```

- b. Mount the USB drive and copy the kernel and the altered `initramfs` into the drive (assuming that the kernel is named `vmlinuz-2.6.18-4-686`):

```
mount -t vfat /dev/sda1 /usbdrive
cp /boot/vmlinuz-2.6.18-4.686 /usbdrive/vml2618
cp /boot/initrd.img-2.6.18-4.686-new /usbdrive/ird2618
```

- c. Create a `syslinux.cfg` file with the following contents inside the drive.

The `vga=791` part is needed to ensure proper X operation:

```
default debilo
label debilo
kernel vml2618
append initrd=ird2618 vga=791
```

- d. Unmount the `usbdrive` and restart the computer.

By now, the computer is able to successfully load Linux off the Loopback Root Filesystem. Still, it is not yet safe to shutdown or reboot the computer without further setup explained in the next subsection.

3.1.3 Further setup to ensure that the Linux Loopback Root Filesystem will shutdown/reboot safely

After setting up the `initramfs`, we need to setup some tasks for the `/sbin/init` process to handle before our system can be shutdown/rebooted. This includes the following tasks involving the scripts listed in Appendix B for the old version and Appendix C for the new version:

1. Creating the `/etc/init.d/debilo_begin` script and the proper symbolic links (applies to both old and new versions, except that the `debilo_begin` contents are different):

```
cd /etc/rcS.d/
ln -s ../init.d/debilo_begin S60debilo_begin
```

2. For the old version, the following applies:
 - a. creating the `/etc/init.d/debilo_end` file and the proper symbolic links

```
cd /etc/rc6.d/  
ln -s ../init.d/debilo_end S70debilo_end  
cd /etc/rc0.d/  
ln -s ../init.d/debilo_end S70debilo_end
```

- b. creating the file `/etc/init.d/ontmpfs`
3. For the new version, the following three files are created:
 - a. `/etc/init.d/ontmpfs` (completely different from 2.b. above)
 - b. `/etc/init.d/induce_err`
 - c. `/etc/nextstep`

When all these files are put in place, the system should be able to startup and shutdown/reboot properly.

3.1.4 Additional customizations to improve the Linux Loopback Root Filesystem

Aside from the previous integral tasks, several other packages can also be installed to improve the user experience on Linux. Aptitude, the APT package management tool front-end, provides users with the option to install many packages through a menu system. It also makes it easy for users to manage installation and removal of package dependencies, whereby any automatically installed packages will be removed as soon as no other packages depend on it. The installation of some tools enabled us to create CD image for use to install the system at the lab.

3.2 Physical setup of a network traffic auditing system

The physical setup of Argus to run on Linux and function as a network traffic auditing system involves two things:

1. Install Linux on one of the computers at the Data Communications and Networking Lab (23-02-13) and setup Argus to run on the computer.
2. Connect several computers to a hub and connect the computer running Argus to the hub. Connect the hub to the UTP network.

3.2.1 *Installation of Linux and Argus*

The only thing needed to install Linux on a computer at the lab is a bootable CD which loads a loopback root filesystem from a hard disk partition. Before beginning this project, such a CD has been prepared to run on a different computer. The only thing needed is to decompress an archived file from the CD, which contains the root filesystem image, into one of the partitions on the computer running Windows.

After the file is created on the partition, the Administrator user of Windows XP has to force the Windows program CHKDSK.EXE to check the integrity of the partition upon next boot up. For example, if the new file is created in C:\, issuing '`CHKDSK /R C:`' will ensure that the C: drive will be checked at the next reboot. Only after CHKDSK is run should Linux be booted.

This is because to run Linux from a loopback root filesystem, Linux need to have read/write access to the file. This is only permitted when Linux has read/write access to the filesystem where the file resides. Most installations of Windows XP usually use the NTFS filesystem, and Linux read/write access to this filesystem is only safe provided that the NTFS filesystem is free of errors [10]. Thus, using CHKDSK functions to ensure that Linux will be able to boot the loopback root filesystem from the NTFS partition.

To boot up Linux, the bootable CD is inserted into the CD-ROM drive. The computer should boot from this drive and load the Linux kernel and the accompanying initramfs (initial ram filesystem). The initramfs contains instructions to the kernel to setup many things before the Linux distribution is ready to be used. In our case, this includes scanning the partitions available on the computer, looking for a file with a particular name pattern (`debian*.img`) and verifying if read/write access to this file is permitted. If this is successful, the file found will be used as the loopback root filesystem.

After Linux has finished booting and the necessary network setup is in place, the user would only need to log into the system and install Argus. As mentioned, this is done by executing the command '`aptitude install argus-server argus-client`' from the Linux login terminal.

3.2.2 Hub connection of Argus to the UTP network

For this setup, we connected several computers running Windows XP to an Ethernet hub. We also connected the Argus computer to the hub, and connect the hub to the wall socket. We did this because we want to know what Argus sees of other hosts' network traffic, which reside on the same network segment. The setup is shown in Figure 1.

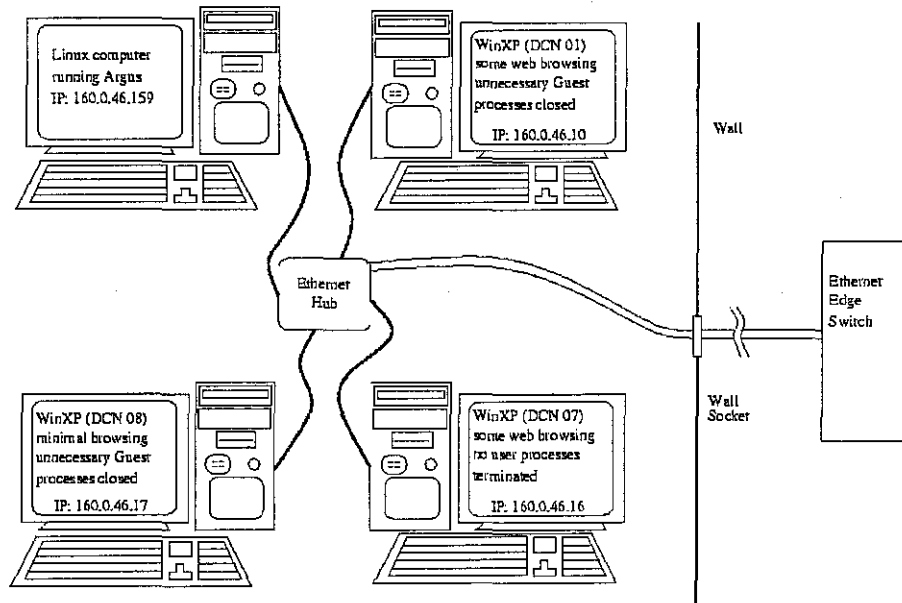


Figure 1 Multiple computers connected to a hub, which connects to the UTP network

The following sections detail how we will use Argus to observe traffic and the accompanying `ra*` programs to parse the Argus-generated data.

3.3 Observing traffic with Argus

For this part of the project, we are interested at looking at all the IP-related traffic that can be read by the computer running Argus. We used the following syntax:

```
argus -m -w /tmp/argus.out - ip
```

which generates all IP and non-MAC statistics detected by Argus, and write them in the file `argus.out` in the `/tmp/` directory. We did this once for the direct connection of Argus to the network, and twice for the hub configuration of Argus.

3.4 Parsing Argus generated data with ra* programs

ra* programs are used to read Argus generated data. Which data to be processed can be specified, similar to the syntax use with Argus above. The following ra* programs are used to parse `argus.out` files generated:

1. `ramon`, to obtain an ordered list of ports used from the network traffic observed, according to the number of packets and the number of bytes used.
2. `ragraph`, to graph the data collected over a period of time according to some defined criteria.
3. `ra`, to apply filters to limit the kind of traffic we want to count using `ramon`.

3.5 An approach to internet bandwidth monitoring

To decide what information is worth for examination, we use `ramon` to parse the `argus.out` files to get a list of most used internet service:

```
ramon -M Svc -N 10 -p0 -n -r argus.out
```

Next, we employ `ra` and `ramon` to obtain a list of the most active hosts on our network, according to the most used service (which is obviously WWW access) above:

```
ra -w - -r argus.out - dst port 80 | ramon net 160.0.46.0/24 -M TopN -p0
```

After that, we use `ragraph` to graph bandwidth usage for every host on our small network, by using the scripts in Appendix D.

3.6 Usage of Argus to monitor the traffic at Village 2 Block E

Initially, the intent to monitor the traffic at Village 2 Block E is done by asking ITMS to mirror the building's edge switch port to the rest of the network to one port we can access. To anticipate for the possibility of a higher number of records generated, we have decided to add the following customizations to the system:

1. Create an empty ext3 filesystem image of size 8GB on a NTFS partition. This is used to store Argus captured network flows.
2. Create a swap file area of size 2GB on a FAT32 partition. This would be used for caching in case Argus needed to use more space in RAM.

As the request to mirror the port is done quite late, the only traffic the machine connected to the port was able to intercept passively was the typical traffic a normal port receives. Although this in itself is not interesting, an analysis of the traffic received poses new questions that we never asked before, and worth discussing in the next chapter. The method used involves the following criteria:

1. Use ra to parse the Argus network flow capture file.
2. Keep only UDP and TCP flow records captured between 11 am and 5 pm from Monday to Thursday (May 5 to May 8, 2008).
3. Filter out all multicast traffic (multicast traffic falls within the IP ranges of 224.0.0.0 to 239.255.255.255, according to RFC 1112--Host extensions for IP multicasting).
4. Filter out all ether broadcast traffic (which includes traffic destined to the FF:FF:FF:FF:FF:FF hardware address, and IP addresses that ends with .255, thus, there is some overlap between multicast and ether broadcast traffic).
5. We also noticed DHCP traffic, thus we filter out all records with destination port 67.
6. During capture, the computer itself may generate traffic, thus we also need to filter out its traffic. This is done by filtering out the IPs used by the computer during these periods.

The result from applying the filters above are stored in a new Argus record file. As we are connected to a switched network, the steps above should have reduced the record of flows into zero or a few records that we may have overlooked. Interestingly, we found that some flows that we would not be able to observe can be seen, while they are clearly meant to be traffic to other computers connected to the switch. We generated graphs of the traffic over a period of time. We discuss this further in the next chapter.

CHAPTER 4

RESULTS AND DISCUSSION

The information in Table 2 details information regarding Argus network traffic capture activities at the DCN lab and the files they generate. We must look at the network services utilized by the computers connected to the network by analyzing these files.

Table 2 Details of Argus network traffic capture activities

Day and Date	Filename	Filesize (bytes)
Wednesday, 4 April 2007	WedApr4-2007	1,083,328
Thursday, 5 April 2007	ThuApr5-2007	1,217,688

The following table shows the 10 most popular traffic detected by ramon.

Table 3 Ten most popular traffic according to ramon.

Port & Protocol	Packets sent	Packets received	Bytes sent	Bytes received
tcp 80	116566	124436	9180784	163724162
tcp 9100	29535	1919	44408167	103714
udp 137	16412	11	1530754	1144
udp 520	4584	0	2063184	0
udp 138	2817	0	694126	0
udp 3024	2381	6	385722	360
icmp	2378	0	451460	0
udp 111	1706	0	235428	0
udp 53	673	667	56266	298007
tcp 139	415	375	58940	45484

The following tables sort the hosts on our network based on the user with the highest utilization of port 80 (www) traffic, using ra and ramon to produce these results.

Table 4 Port 80 traffic contribution sorted by most generated to least generated for Wednesday April 4th 2007

IP address	Packets received	Packets sent	Bytes received	Bytes sent
160.0.46.159	32888	32004	47853110	1877863
160.0.46.10	15310	12399	11218488	1821616
160.0.46.17	328	1716	281812	122063
160.0.46.16	513	627	380947	64897

Table 5 Port 80 traffic contribution sorted by most generated to least generated for Thursday April 5th 2007

IP address	Packets received	Packets sent	Bytes received	Bytes sent
160.0.46.159	64763	59025	94285317	4061355
160.0.46.10	8188	6483	7346830	893438
160.0.46.16	2176	1916	2089503	177538
160.0.46.17	270	2395	268155	161960

4.1 Internet bandwidth monitoring results

Some of the graphs generated using the scripts in Appendix D can be seen in Figure 2. We observed that the time taken to generate 1-second average graphs is much longer than 5-minute average graphs, as the finer grained statistics in 1-second averages takes longer to calculate compared to 5-minute average calculations.

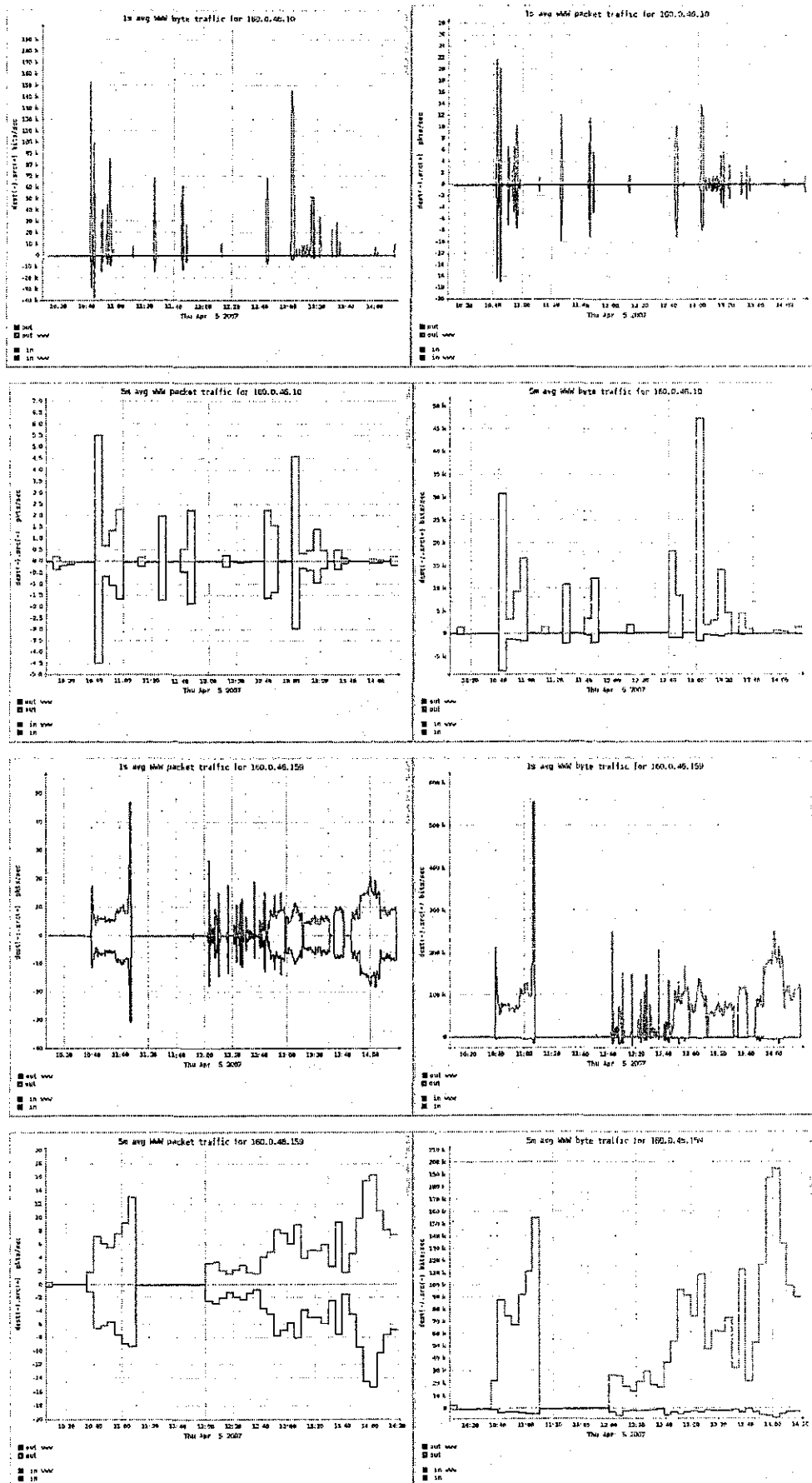


Figure 2 Several graphs generated by the scripts in Appendix D

4.2 Results of the Village 2 Block E normal traffic observations with filters applied

The following graphs illustrate the bits and packets data 1-second average registered by Argus and applied with filters, for packets not belonging to the receiving port.

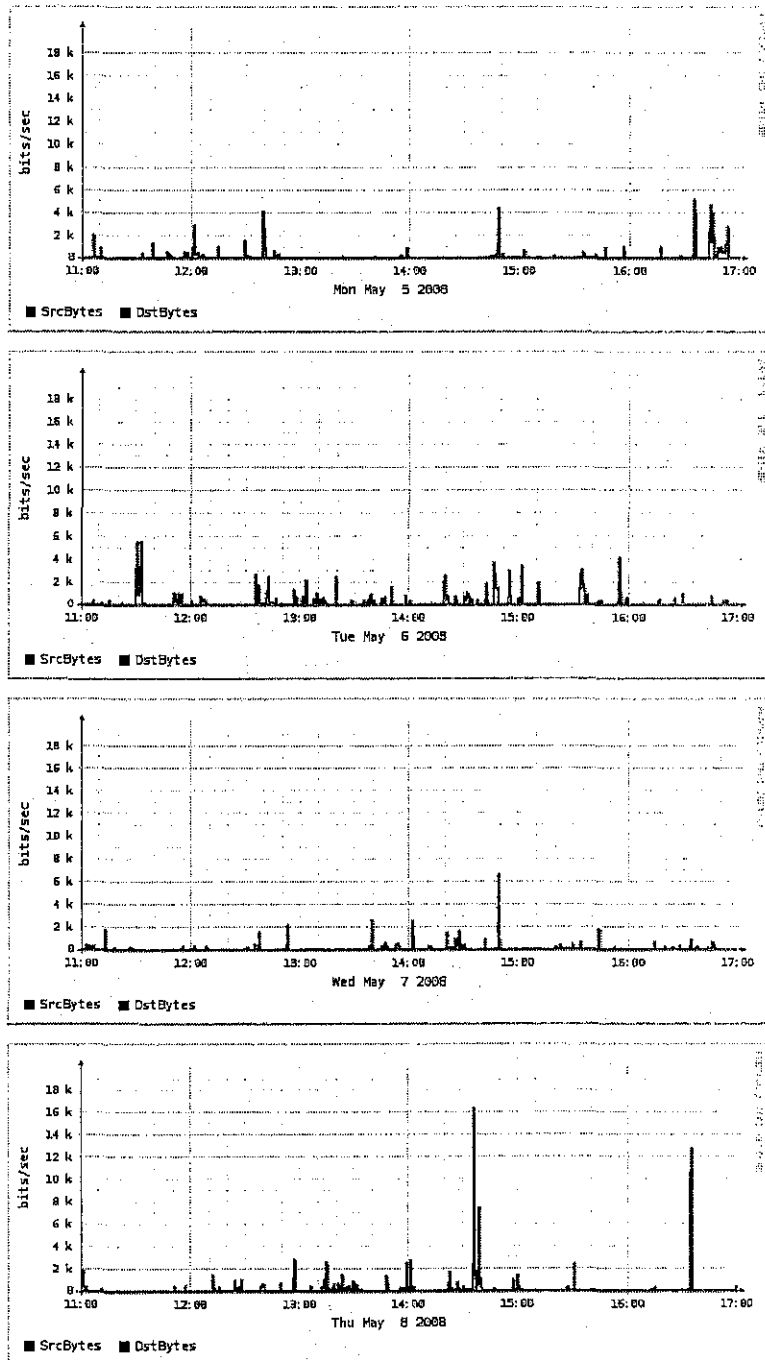


Figure 3 Graphs showing rate of bits received on the V2E network port that do not belong to the port, between 11 am and 5 pm for May 5 – 8, 2008.

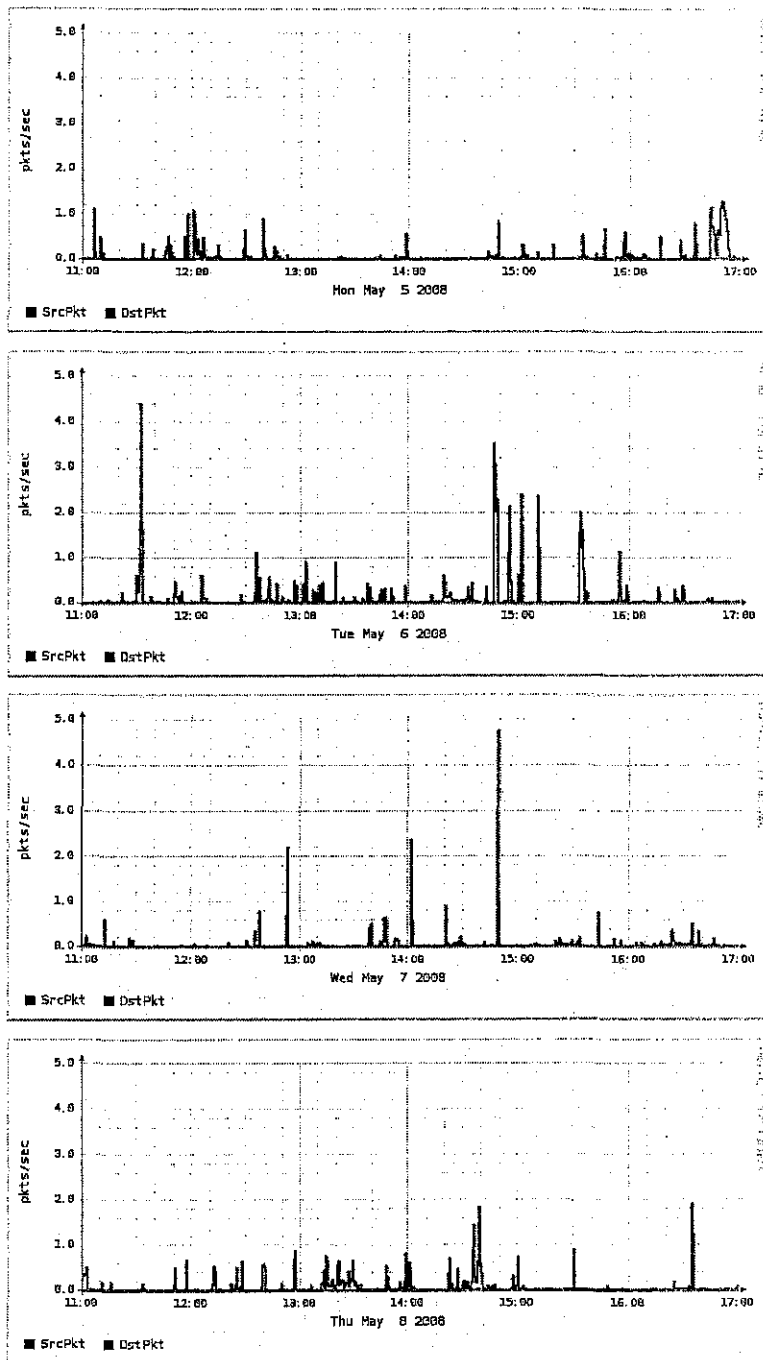


Figure 4 Graphs showing rate of packets received on the V2E network port that do not belong to the port, between 11 am and 5 pm for May 5 – 8, 2008.

Although this phenomenon may seem peculiar at first, the reason for the switch to behave in such a way may be to avoid from dropping these packets while sorting them out to the proper ports. It is more beneficial to the switch to just broadcast these packets, which will be received by the intended recipients anyway.

It could also be a sign that the switch is overloaded while handling the number of packets that is flowing through it. Still, this issue has to be investigated further before any conclusions can be made about it.

4.3 Drawbacks of the Linux Loopback Root Filesystem as a network traffic auditing system

As can be seen, even traffic contributed mostly by four computers over a span of about four hours generate an Argus log of about 1MB in size. As the size of the whole Loopback Root Filesystem is smaller than typical hard disk size nowadays, if this was used on an active network, the disk space might be easily used up over a span of a few hours.

Also, as the NTFS support of the Linux kernel is not robust against forced umounts, each time a failure which forces the user to run CHKDSK.EXE means some downtime to the system, thus reducing its usefulness as a network traffic auditing system.

This was a problem for the setup used at Village 2 Block E. As the system did not know how to automatically umount the 8GB ext3 filesystem image on the NTFS partition before unmounting the NTFS partition, the system restarted with the NTFS partition marked for disk checking, and mounted read-only by Linux.

4.4 How to address the drawbacks of the system

Instead of looking at the current system as the ends for a network traffic auditing system, it should be seen as a means towards implementing a full-fledged system. For example, in the early stages this system can be used to randomly observe traffic at several different spots on the network. Its portability is a strength that most systems may not offer. After identifying metrics for measure, a more permanent setup of Debian can be deployed. This can be a simple act of copying the Loopback Root Filesystem to an empty ext3 partition. This solution above should address drawbacks introduced by the NTFS driver.

In the meantime, to increase the space of the current Loopback Root Filesystem

without creating an empty ext3 partition, the following can be done:

1. From within Windows, make a copy of the Loopback Root Filesystem (for example, copy `debianS0.img` to `copydeb.img`).
2. Create a file of new target size for the new Loopback Root Filesystem (for example, create a 10GB file with name `newdeb.img` on an NTFS partition). We can use Cygwin's `dd.exe` on Windows to create this file.
3. After creating the file, we boot into Linux, and mount the partition containing `newdeb.img` and format the `newdeb.img` file as an ext3 filesystem.
4. Mount the `copydeb.img` file and the `newdeb.img` file, (e.g., to the directories `/copydeb/` and `/newdeb/`, respectively.)
5. Copy the whole directory tree structure from `/copydeb/` to `/newdeb/`, (e.g., `cd /copydeb/ && cp -a [a-k]* lib [m-z]* -t /newdeb/`, this is done as such to skip copying the `lost+found` directory).
6. Umount `copydeb.img` and `newdeb.img`
7. Run windows, remove old debian system and its copy (`debianS0.img` and `copydeb.img`) and rename `newdeb.img` into `debianS0.img`.

To skip step 1, the user may opt to boot from a Linux Live-CD, such as Knoppix, instead of running Linux from the Loopback Root Filesystem, and mount `debianS0.img` and copy its directory tree structure instead.

4.5 Other benefits of the system

Although we only use this system for network traffic auditing, Linux is very capable at handling many different tasks. By using aptitude, users can try different packages related to their field and interests. All this can be done simply by copying the Loopback Root Filesystem to a computer's partition and booting the system from the CDROM drive.

4.6 Some thoughts regarding the progression of the project

Since the beginning of the January 2008 semester, my supervisor has contacted ITMS regarding this project, by showing what it has achieved so far, and what needs to be done in the future. I am grateful that I could finally speak with several IT executives some time in the middle of the semester to talk about the progression of the project.

One of the things mentioned was the need for me to use an Ethernet port at my dorm room as the mirror port for the network traffic auditing system. Realizing that I have to forgo my network access for the sake of this project, I have decided to push the testing of the system towards the end of the semester. What I fail to realize, however, is that to get ITMS to mirror the port would also require a considerable amount of time. It is my opinion that our IT executives have a lot of responsibilities to handle and there are not enough staff to cater to some needs that need to be fulfilled in the near term.

Although this has affected the project this semester, the project itself is not a total loss. It has been able to run Linux and function as a network traffic auditing system, even if it has yet to be tested with real network traffic. In fact, we would not realize that the V2E switch broadcasts some packets instead of processing each individual packets before transmitting it to the proper port, and we are able to highlight this issue for further study.

4.7 Outlook on future studies of network traffic auditing

If we are to look at the current state of this study, we can be assured that it can continue. But the scope at which it should be applied needs to be framed on its usefulness on the UTP network. This depends on two things:

1. Change in the management's view of the UTP network.
2. Making the IT administration of the network more robust.

4.7.1 Change in the management's view of the UTP network

The need to change the management's view of the UTP network hinges on the fact that without the change, the problems faced by the UTP network will not be fixed.

As long as students claim that they are facing problems with internet access not being worth as much as the networking services fees they have to pay, the management's reply is that the users need to change their use of the network to focus on academic pursuits, which will then solve the problem. The problem with this line of argument is that both sides expect the other to take action, when none would be willing or able to make a change.

The question of changing the management's view of the network lies with changing the context of the situation. Instead of the UTP network existing for "academic purposes", wouldn't it be more natural for the UTP network to exist for the students to participate in a global internet? After all, outside of UTP, their access to the internet would not be as limited as it is inside. Shouldn't the experience inside the UTP network would closely resemble what they have outside?

Admittedly, the limits are imposed because of the problems of not putting limits making the whole user experience on the network more unbearable. The question is, if the limits are not preferable in the first place, shouldn't we be seeking favorable alternatives by now? Of course, if the limits put are necessary, we should not remove them until a better alternative existed.

So, the debate should be framed towards making the students be prepared to what they have outside of UTP, compared to what is deemed "proper" inside UTP. With the change in the management's view, this would also call for the need to improve the IT administration.

4.7.2 Making the IT administration of the network more robust

As it stands, ITMS needs to make changes to improve the administration of the network. Based on my limited experience of getting in touch with ITMS, it is very hard to get things done in a short amount of time. The problem is that ITMS staff

faces too many responsibilities at a time for a small number of people. Thus, considering that there would be more important things to handle, I found the requests I made put in a low priority.

I would suggest ITMS to open up to UTP users not only when they come to have their problems solved, but also to listen and take action when they propose a solution to the problem. The key to taking action on the proposal is by appointing any staff available and to give the staff overseer status of the problem area. This shows that ITMS can accommodate users, and is willing to take the time to fix the problem.

Is it necessary for the ITMS staff to be very knowledgeable about the problem area? Although it would be beneficial to have knowledgeable staff in charge, a staff with the very basic knowledge about the problem and the willingness to try solving the problem should be enough. After all, we expect our students to not know everything but have the capacity to find solutions creatively, why not we find the same from our ITMS staff?

At the same time, instead of asking ITMS to do something to improve the network, the students themselves should propose these changes. The first step that they must take is finding a more knowledgeable person to guide them in their pursuits. Some of our EE and ICT lecturers are very knowledgeable of the underlying data and communication network systems in use at UTP and are always ready to guide students who need their help. Students who are interested in improving the UTP network should approach these lecturers and tell them the situation regarding the network that they believe needs to improve. Students must also be willing to work independently, and try to find like-minded individuals to help them in pursuit of solving the problem.

With those thoughts in mind, not only does this project can be improved, but any project to improve the condition of the UTP network would be welcomed.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

Network traffic auditing has many applications, ranging from fair user bandwidth allocation to network traffic discrimination. The field which network traffic auditing applies depend on the technology used.

Network flow analysis as one form of network traffic auditing has been used at several universities to enforce limits on the fair use of the internet bandwidth. Although the same issues may not apply to UTP, the application of network flow analysis may be useful in another area.

To make deployment of a network traffic auditing system simple, we have embarked upon building Debian GNU/Linux on Loopback Root Filesystem. The system works well and needs to be improved in several areas. Aside from its use as a network traffic auditing system, it can also be used as a general purpose Debian GNU/Linux operating system. Still, it needs to be tested on a real network to be tested as a network traffic system.

The outlook for more research projects to improve the UTP network would depend largely on UTP management's need to change its views of the UTP network and the IT administration's need to be more robust at managing the network. These goals can only be achieved with the cooperation from ITMS, the students and EE and ICT faculty members.

REFERENCES

- [1] W. Stallings, *Data & Computer Communications*, Seventh Edition, Prentice Hall, 2004.
- [2] Argus – Home, Qosient, LLC, “Argus - Manuals,” May 1, 2008. <http://qosient.com/argus/ra.1.htm> .
- [3] Wireshark: Go deep., Gerald Combs, “Wireshark FAQ,” May 1, 2008. <http://www.wireshark.org/faq.html> .
- [4] J. Peha, “The Benefits and Risks of Mandating Network Neutrality and the Quest for a Balanced Policy,” presented at the 34th Research Conference on Communication, Information, and Internet Policy, Telecommunications Policy Research Conference, George Mason University, VA, 2006.
- [5] W.H. Lehr, S.E. Gillette, M.A. Sirbu, and J. Peha, “Scenarios for the Network Neutrality Arms Race,” presented at the 34th Research Conference on Communication, Information, and Internet Policy, Telecommunications Policy Research Conference, George Mason University, VA, 2006.
- [6] Free Press Action Fund (caaron@freepress.net) “Save the Internet coalition,” March 24, 2008. <http://www.savetheinternet.com/=coalition> .
- [7] Hands off the Internet (info@handsoff.org) “Hands off the Internet » About Us,” March 24, 2008. <http://www.handsoff.org/blog/about-us> .
- [8] P. Hill and K. Miller, “Successful Bandwidth Management at Carnegie Mellon,” presented at Internet2 Joint Techs Meeting, Lawrence, KS, 2003.
- [9] myBrandeis Technical Support forum, Brandeis University, “The Network Bandwidth Thread that Ate Sheboygan,” September 13, 2005. Begins at http://my.brandeis.edu/bboard/q-and-a-fetch-msg?msg_id=00008k .
- [10] R. Watt, Information Systems and Technology, University of Waterloo, “Coping with the impact of networked residences at the University of Waterloo,” September 13, 2005. <http://ist.uwaterloo.ca/cn/Residence/history.html> .
- [11] G. Ray, “Quality of Service in Data Networks: Products,” student project report for CIS 788.08Q, Department of Computer Science and Engineering, Washington University in St. Louis, Autumn 1999.
- [12] Debian, Software in the Public Interest, Inc., “Debian -- The Universal Operating System,” January 15, 2007. <http://www.debian.org> .
- [13] A. Patyk (areq@pld-linux.org), “PLD RescueCD,” February 15, 2007. <http://rescuecd.pld-linux.org/> .
- [14] K. Knopper (knoppix@knopper.net), “KNOPPIX,” March 21, 2005. <http://knopper.net/knoppix/index-en.html> .
- [15] A. M. Bishop (amb@gedanken.demon.co.uk), “The Loopback Root Filesystem HOWTO,” February 14, 2007. <http://tldp.org/HOWTO/archived/Loopback-Root-FS/> .
- [16] Pigeon (pigeon@pigeond.net), “Pikkoro - Episode 2 - Loopback rootfs with Debian,” February 14, 2007. <http://pigeond.net/index.html?cat=linux&comments=pikkoro2> .

- [17] Linux-NTFS-Project, "Linux-NTFS kernel driver Roadmap," February 14, 2007. <http://wiki.linux-ntfs.org/doku.php?id=driverroadmap> .
- [18] Argus – Home, Qosient, LLC, "Argus Auditing Network Activity," February 10, 2007. <http://qosient.com/argus/index.htm> .
- [19] News@Cisco, Cisco Systems Inc., "News @ Cisco: Routing GLOSSARY," March 27, 2007. http://newsroom.cisco.com/dlls/2004/hd_051904c.html .
- [20] Meeting with UTP IT and Media Services executives, Mr. Arfaishah and Mr. Abdul Rahim, on March 13, 2008.
- [21] Datuk Dr. Zainal Abidin Hj. Kasim, "Latest News from Rector Jan 2008, 5th March 2008 posting, UTP Chat - Internet Relay Chat (IRC)," March 20, 2008. <http://elearning.utp.edu.my/course/view.php?id=333> .

APPENDICES

APPENDIX A
CUSTOMIZED INITRAMFS

```
#####  
# file: /etc/initramfs-tools/modules  
# List of modules that you want to include in your initramfs.  
#  
# Syntax:  module_name [args ...]  
#  
# You must run update-initramfs(8) to effect this change.  
#  
# Examples:  
#  
# raid1  
# sd_mod  
loop  
vfat  
ntfs  
nls_cp437  
nls_iso8859_1  
nls_utf8  
#####  
# file: /etc/initramfs-tools/conf.d/root  
ROOT=/dev/loop0  
#####
```



```

#####
# file: /etc/initramfs-tools/scripts/debilo_local
# -*- scripts/debilo_local shell-script -*-
# helper functions used by other scripts to prepare
# a debian loopback image file on a real partition.
# (C) 2006 Ghauth Hassan (ghauth_at_yahoo_dot_com)
# licensed under the GNU GPL (http://www.gnu.org/licenses/gpl.html)

show_link_deps() {
    # shows us the files needed off of ${rootmnt} to run $1
    echo ${rootmnt}${1}
    ${rootmnt}/lib/ld-linux.so.2 --list --library-path ${rootmnt}/lib/ \
        ${rootmnt}${1} | sed '/=> \\/ !d;s,^.* \{/.*\} .*$,\\1,g'
}

copy_files_for() {
    # copies the files needed, to run $1 on ${DEBILO_DIR}
    FILES_NEEDED=$(show_link_deps $1)
    # define the special destination file naming operation
    DST_FILE_OP=$(echo "echo \${EACH_FILE} | sed 's,${rootmnt},${DEBILO_DIR},g'")
    for EACH_FILE in ${FILES_NEEDED}; do
        DST_FILE=$(eval "$DST_FILE_OP")
        # if $DST_FILE does not exist, copy file
        [ -f $DST_FILE ] || {
            [ -L $EACH_FILE ] && \
                SRC_FILE=$(dirname $EACH_FILE)/$(readlink $EACH_FILE) || \
                SRC_FILE=$EACH_FILE
            # directory location of $DST_FILE
            DST_DIR=$(dirname $DST_FILE)
            [ -d $DST_DIR ] || mkdir -p $DST_DIR
            cp -p $SRC_FILE $DST_FILE
        }
    done
    return 0
}
#####

```

```
#####
# file: /etc/initramfs-tools/scripts/debilo_init
# -*- scripts/debilo_init shell-script -*-
# helper functions used by other scripts to prepare
# a debian loopback image file on a real partition.
# (C) 2006 Ghauth Hassan (ghauth_at_yahoo_dot_com)
# licensed under the GNU GPL (http://www.gnu.org/licenses/gpl.html)

unsorted_part_ls() {
    for i in /dev/.udev/db/block@*?*; do
        sed -ne '/^N\|FS_USAGE\|FS_TYPE\|FS_VERSION/ H
        $ {
            g
            s/\nN:\(...\)\{.*\}\n.*=filesystem\n.*=\{.*\}\n.*=\{.*\}/\2 \1\2 \3 \4/p
        }' $i
    done
}

get_part_ls() {
    # get a list of partition, fs type and version
    # final output looks similar to this:
    # hda1 ntfs 3.1
    # hda2 vfat FAT32
    # hdb1 ext3 1.0
    # sda1 vfat FAT16
    TMP_PART_LS=$(unsorted_part_ls)
    TMP_BLK_LS=$(ls -l /dev/.udev/db/block@??? | sed -ne 's/.*\(...\)$/\1/p')
    for i in ${TMP_BLK_LS}; do
        echo "${TMP_PART_LS}" | grep .* $i | sort -n | cut -d\ -f2,3,4
    done
}

verify_host() {
    echo -n "Checking ${HOSTFS} for read-write access .. "
    grep -q "^${HOSTFS} ${HOST_MNT} ${HOST_TYPE} rw" /proc/mounts && \
        echo OK || {
            echo -en "No read-write access!\nUnmounting ${HOSTFS} .. "
            umount ${HOST_MNT} && echo Done || echo Failed
            echo -n "If you are getting this error, you might need to run"
            echo "'chkdsk' on MS Windows to fix this problem"
            return 1
        }
    return 0
}

prep_host() {
    # prepare HOSTFS for use
    # check for fstype, load module
    HOST_TYPE=$(echo "${PARTLS}" | grep ^`basename ${HOSTFS}`\ | cut -d\ -f2)
    modprobe -q ${HOST_TYPE} 2> /dev/null
}

```

```

# additional mount options
[ -z "${HOST_OPT}" ] && {
    HOST_OPT="defaults"
    [ ${HOST_TYPE} = "ntfs" ] && \
        HOST_OPT="uid=1000,gid=1000,umask=022,nls=utf8"
    [ ${HOST_TYPE} = "vfat" ] && \
        HOST_OPT="uid=1000,gid=1000,umask=022,nonumtail,utf8"
}
# mount HOSTFS
echo -n "Mounting ${HOSTFS} .. "
/bin/mount -t ${HOST_TYPE} -o ${HOST_OPT} ${HOSTFS} ${HOST_MNT} \
    2> /dev/null && echo Done || { echo Failed; return 1; }
verify_host || return 1
}

prep_img() {
    # prepare IMGNAME for use
    # check if it is a valid filesystem
    /lib/udev/vol_id ${HOST_MNT}/${IMGNAME} 2> /dev/null | \
        grep -q ^ID_FS_USAGE=filesystem && continue || {
            echo -n "${IMGNAME} is not a valid filesystem. Unmounting ${HOSTFS} .. "
            umount ${HOST_MNT} && echo Done || echo Failed
            return 1
        }
    echo -n "Preparing loopback root filesystem image (${IMGNAME}) as /dev/loop0 .. "
    losetup /dev/loop0 ${HOST_MNT}/${IMGNAME} && echo Done || \
        { echo Failed; losetup -d /dev/loop0 2> /dev/null; return 1; }
    return 0
}

scan_img() {
    # scans for debian loopback root filesystem images
    echo "${PARTLS}" | while read PART FSTYPE FSVER; do
        modprobe -q ${FSTYPE} 2> /dev/null
        /bin/mount -r -t ${FSTYPE} /dev/${PART} ${HOST_MNT}
        for i in $(ls -l ${HOST_MNT}/debian*.img 2> /dev/null); do
            echo "HOSTFS=/dev/${PART}; IMGNAME=$(basename $i)"
            done
        umount ${HOST_MNT}
        modprobe -rq ${FSTYPE} 2> /dev/null
    done
}

clr_line() {
    # clears the line for a new line
    j=$(length "${TMP_STRING}")
    while [ $j -ge 0 ]; do echo -en '\b \b'; j=$((j-1)); done
}

```

```

countdown() {
    # a fancy countdown counter: countdown X [prepending sentence], X in seconds
    # will print as follows: prepending sentence X seconds.
    # if there are no arguments, will print as follows: Rebooting in X seconds.
    # numbers are approximate :P
    i=$1
    shift
    [ "${#0}" -eq 0 ] && echo -n "Rebooting in " || echo -n "$0 "
    while [ $i -ge 1 ]; do
        [ $i -gt 1 ] && TMP_STRING="$i seconds." || TMP_STRING="$i second. "
        echo -n ${TMP_STRING}\
        sleep 1
        clr_line
        i=$((i-1))
    done
    clr_line; clr_line
    return 0
}
#####

```

```
#####
# file: /etc/initramfs-tools/scripts/init-premount/debilo
#!/bin/sh -e
# -*- scripts/init-premount/debilo -*-
# prepare a debian loopback image file on a real partition.
# (C) 2006 Ghauth Hassan (ghauth_at_yahoo_dot_com)
# licensed under the GNU GPL (http://www.gnu.org/licenses/gpl.html)
PREREQ="udev"
prereqs()
{
    echo "$PREREQ"
}

case $1 in
# get pre-requisites
prereqs)
    prereqs
    exit 0
    ;;
esac

# check if root device is either "/dev/loop0" or "7:0" to continue
[ "${ROOT}" = "7:0" -o "${ROOT}" = "/dev/loop0" ] && continue || exit 0

# source functions for use later
. /scripts/debilo_init

# define variable for the host filesystem mountpoint
[ -z ${HOST_MNT} ] && export HOST_MNT=/dev/.host

# keep info for host filesystem mountpoint for access later
echo HOST_MNT=${HOST_MNT} > /dev/.initramfs_debilo

# create the $HOST_MNT directory
mkdir -p ${HOST_MNT}

# Parse command line options
export HOSTFS=
export HOST_OPT=
export IMGNAME=
```

```

for x in $(cat /proc/cmdline); do
  case $x in
    hostfs=*)
      HOSTFS=${x#hostfs=}
      ;;
    host_opt=*)
      HOST_OPT=${x#host_opt=}
      ;;
    imgname=*)
      IMGNAME=${x#imgname=}
      ;;
  esac
done

# define a list of partitions
PARTLS=$(get_part_ls)

# if HOSTFS is a valid block device, setup HOSTFS for use
[ -b "${HOSTFS}" ] && prep_host

# if IMGNAME is a valid file, setup IMGNAME for use
[ -f "${HOST_MNT}/${IMGNAME}" ] && prep_img || IMGNAME=''

# if we have both set by now, IMGNAME will represent the loopback root
# filesystem that has been setup, so we just exit now
[ -z "${IMGNAME}" ] && continue || exit 0

# if we reach here, we don't have any IMGNAME detected
echo -e "Scanning for valid debian loopback root filesystem images.
Here is a list of hostfs=* imgname=* pair(s), if any exists.
We will try to setup the first three pairs, but stop trying
when we find a usable pair or we fail on the third try.\n"

# get a list of HOSTFS=*; IMGNAME=* pairs, print the list
HOST_IMG_PAIRS=$(scan_img)
echo "${HOST_IMG_PAIRS}" | \
  sed 's/^\(HOSTFS\) \([^\; ]*\) \({; }\) \(\(IMGNAME\) \(.*)\)$ /hostfs\2 imgname\5/g'

# initially, we do not have the loopback image set
LOOP_IMG_SET="no"

# if the pairs list is non-empty, note number of lines of pairs
[ -z "${HOST_IMG_PAIRS}" ] && PAIR_TOT=0 || \
  PAIR_TOT=$(echo "${HOST_IMG_PAIRS}" | wc -l)

```

```

# find valid image routine, done at most three times

[ ${PAIR_TOT} -ge 1 ] && {
    # unset HOST_OPT, as this screws up our later mounts
    HOST_OPT=''
    echo -e "\nFirst Try ..\n"
    eval $(echo "${HOST_IMG_PAIRS}" | tail -n+1 | head -1)
    [ -b "${HOSTFS}" ] && prep_host
    [ -f ${HOST_MNT}/${IMGNAME} ] && prep_img && LOOP_IMG_SET="yes"
    echo
}

[ ${PAIR_TOT} -ge 2 -a "${LOOP_IMG_SET}" = "no" ] && {
    # unset HOST_OPT, as this screws up our later mounts
    HOST_OPT=''
    echo -e "Second Try ..\n"
    eval $(echo "${HOST_IMG_PAIRS}" | tail -n+2 | head -1)
    [ -b "${HOSTFS}" ] && prep_host
    [ -f ${HOST_MNT}/${IMGNAME} ] && prep_img && LOOP_IMG_SET="yes"
    echo
}

[ ${PAIR_TOT} -ge 3 -a "${LOOP_IMG_SET}" = "no" ] && {
    # unset HOST_OPT, as this screws up our later mounts
    HOST_OPT=''
    echo -e "Third Try ..\n"
    eval $(echo "${HOST_IMG_PAIRS}" | tail -n+3 | head -1)
    [ -b "${HOSTFS}" ] && prep_host
    [ -f ${HOST_MNT}/${IMGNAME} ] && prep_img && LOOP_IMG_SET="yes"
    echo
}

# by now, we should have a loopback image set, else we restart the computer
[ "${LOOP_IMG_SET}" = "yes" ] && countdown 10 Continuing in && exit 0

echo "It seems there is no valid debian loopback root filesystem images
available! Press enter to reboot."
read XXXX

# will reboot after about 5 seconds
countdown 5 && reboot
#####

```

```
#####
# file: /etc/initramfs-tools/scripts/local-bottom/debilo
#!/bin/sh -e
# -*- scripts/local-bottom/debilo -*-
# prepare a debian loopback image file on a real partition.
# (C) 2006 Ghauth Hassan (ghauth_at_yahoo_dot_com)
# licensed under the GNU GPL (http://www.gnu.org/licenses/gpl.html)
PREREQ=""

prereqs() { echo "$PREREQ"; }

case "$1" in
    prereqs)
        prereqs
    exit 0
    ;;
esac

# define variable for the host filesystem mountpoint
[ -z ${HOST_MNT} ] && export HOST_MNT=/dev/.host

# check if ${HOST_MNT} exists and is a directory, else exit now
[ -d ${HOST_MNT} ] 2> /dev/null && continue || exit 0

# we just assume that when the above exists, we are running a loopback rootfs

# source functions for use later
. /scripts/debilo_local

# our debilo directory location (important to export this variable)
[ -z ${DEBILO_DIR} ] && export DEBILO_DIR=/dev/.debilo

# create tmpfs directory
mkdir -p ${DEBILO_DIR}

# our tmpfs options
TMPFS_OPTS="size=5M,mode=0755"

# mount the tmpfs
mount -t tmpfs -o ${TMPFS_OPTS} debilo ${DEBILO_DIR} || exit 0

# keep info for debilo tmpfs mountpoint for access later
echo DEBILO_DIR=${DEBILO_DIR} >> /dev/.initramfs_debilo

# copy the necessary files to run ${init} from ${DEBILO_DIR}
copy_files_for ${init}
```



```

# define the special sed operation
SED_OP=$(echo "sed -e '
s,10:0:wait:/etc/init.d/rc 0,10:0:wait:${DEBILO_DIR}/lib/ld-linux.so.2 --library-path
${DEBILO_DIR}/lib ${DEBILO_DIR}/bin/sh ${DEBILO_DIR}/etc/init.d/rc 0,
s,16:6:wait:/etc/init.d/rc 6,16:6:wait:${DEBILO_DIR}/lib/ld-linux.so.2 --library-path
${DEBILO_DIR}/lib ${DEBILO_DIR}/bin/sh ${DEBILO_DIR}/etc/init.d/rc 6,'"')

# create special inittab file
mkdir ${DEBILO_DIR}/etc
cat ${rootmnt}/etc/inittab | eval "${SED_OP}" > ${DEBILO_DIR}/etc/inittab

# bind ${DEBILO_DIR}/etc/inittab to ${rootmnt}/etc/inittab
mount -n -o bind ${DEBILO_DIR}/etc/inittab ${rootmnt}/etc/inittab

# write a small note inside ${DEBILO_DIR}
echo "If you need to change anything inside this directory, you need
to check how it affects the rc scripts used. Please do not
change anything inside here if you do not know how it will
affect the system." > ${DEBILO_DIR}/readme.txt

# generate proper scripts/initline file
[ -e /scripts/initline ] || \
echo "exec run-init ${rootmnt} ${DEBILO_DIR}/lib/ld-linux.so.2 \\\
--library-path ${DEBILO_DIR}/lib ${DEBILO_DIR}/${init} \\\
\"\\$@\" <${rootmnt}/dev/console >${rootmnt}/dev/console" > /scripts/initline

exit 0

#####

```

```
#####
# file: /etc/initramfs-tools/scripts/init-premount/new-debilo
#!/bin/sh -e
# -*- scripts/local-bottom/new-debilo -*-
# prepare a debian loopback image file on a real partition.
# (C) 2006 Ghauth Hassan (ghauth_at_yahoo_dot_com)
# licensed under the GNU GPL (http://www.gnu.org/licenses/gpl.html)
PREREQ=""

prereqs() { echo "$PREREQ"; }

case "$1" in
    prereqs)
        prereqs
        exit 0
    ;;
esac

# define variable for the host filesystem mountpoint
[ -z ${HOST_MNT} ] && export HOST_MNT=/dev/.host

# check if ${HOST_MNT} exists and is a directory, else exit now
[ -d ${HOST_MNT} ] 2> /dev/null && continue || exit 0

# we just assume that when the above exists, we are running a loopback rootfs

# source functions for use later
. /scripts/debilo_local

# our debilo directory location (important to export this variable)
[ -z ${DEBILO_DIR} ] && export DEBILO_DIR=/dev/.debilo

# create tmpfs directory
mkdir -p ${DEBILO_DIR}

# our tmpfs options
TMPFS_OPTS="size=5M,mode=0755"

# mount the tmpfs
mount -t tmpfs -o ${TMPFS_OPTS} debilo ${DEBILO_DIR} || exit 0

# keep info for debilo tmpfs mountpoint for access later
echo DEBILO_DIR=${DEBILO_DIR} >> /dev/.initramfs_debilo

# copy the necessary files to run ${init} from ${DEBILO_DIR}
copy_files_for ${init}
```

```

# define the special sed operation
SED_OP="sed -e '
s,l6:6:wait:/etc/init.d/rc 6,&\nl7:7:wait:/etc/init.d/rc 7,
s,z6:6:respawn,z7:7:respawn,
s,ca:12345:ctrlaltdel;,&,'"

# create special inittab file
mkdir ${DEBILO_DIR}/etc
cat ${rootmnt}/etc/inittab | eval "${SED_OP}" > ${DEBILO_DIR}/etc/inittab

# bind ${DEBILO_DIR}/etc/inittab to ${rootmnt}/etc/inittab
mount -n -o bind ${DEBILO_DIR}/etc/inittab ${rootmnt}/etc/inittab

# write a small note inside ${DEBILO_DIR}
echo "If you need to change anything inside this directory, you need
to check how it affects the rc scripts used. Please do not
change anything inside here if you do not know how it will
affect the system." > ${DEBILO_DIR}/readme.txt

# generate proper scripts/initline file
[ -e /scripts/initline ] || \
echo "exec run-init ${rootmnt} ${DEBILO_DIR}/lib/ld-linux.so.2 \\\
--library-path ${DEBILO_DIR}/lib ${DEBILO_DIR}/${init} \\\
\"\\$@\" <${rootmnt}/dev/console >${rootmnt}/dev/console" > /scripts/initline

exit 0

#####

```

```
#####
# file: local-bottom_debilo.diff
*** old_local-bottom_debilo   Mon Jan 15 08:04:04 2007
--- new_local-bottom_debilo   Sun Aug 12 23:19:02 2007
*****
*** 44,52 ****
    copy_files_for ${init}

-# define the special sed operation
! SED_OP=$(echo "sed -e '
! s,10:0:wait:/etc/init.d/rc 0,10:0:wait:${DEBILO_DIR}/lib/ld-linux.so.2 --library-
path ${DEBILO_DIR}/lib ${DEBILO_DIR}/bin/sh ${DEBILO_DIR}/etc/init.d/rc 0,
! s,16:6:wait:/etc/init.d/rc 6,16:6:wait:${DEBILO_DIR}/lib/ld-linux.so.2 --library-
path ${DEBILO_DIR}/lib ${DEBILO_DIR}/bin/sh ${DEBILO_DIR}/etc/init.d/rc 6,'"

# create special inittab file
mkdir ${DEBILO_DIR}/etc
--- 44,53 ----
    copy_files_for ${init}

# define the special sed operation
! SED_OP="sed -e '
! s,16:6:wait:/etc/init.d/rc 6,&\n17:7:wait:/etc/init.d/rc 7,
! s,z6:6:respawn,z7:7:respawn,
! s,ca:12345:ctrlaltdel:,&,'"

# create special inittab file
mkdir ${DEBILO_DIR}/etc
#####
```

APPENDIX B
OLD VERSION SCRIPTS

```
#####
# file: /etc/init.d/debilo_begin
#! /bin/sh
### BEGIN INIT INFO
# Provides:          debilo_begin
# Required-Start:    hwclockfirst.sh checkfs.sh
# Required-Stop:
# Should-Start:
# Default-Start:     S
# Default-Stop:
# Short-Description: Continuing to setup debilo for further use.
# Description:       Copying necessary files to ${DEBILO_DIR}, remedy some
#                   issues regarding hwclock, fake mount ${HOST_MNT}
#                   and ${DEBILO_DIR}
### END INIT INFO
PATH="/usr/sbin:/usr/bin:/sbin:/bin"
# some checks that need to be done before we go on
# 1. the /dev/.initramfs_debilo file must exist
[ -f /dev/.initramfs_debilo ] && . /dev/.initramfs_debilo || exit 0
# 2. by sourcing /dev/.initramfs_debilo , we should have ${DEBILO_DIR} set
#    and it should be a mounted directory
[ -d "${DEBILO_DIR}" ] && mountpoint -q "${DEBILO_DIR}" || exit 0
# if we pass the tests, we assume everything is ok
echo "Continuing to setup debilo for use .."
# Executing the necessary tasks
# A-1) Functions used to copy the necessary files
show_link_deps() {
    # shows us the files needed off of / to run $1
    echo ${1}
    /lib/ld-linux.so.2 --list ${1} | sed '/=> \/\/ !d;s,^.* \(/.*\) .*$,\\1,g'
}
copy_files_for() {
    # copies the files needed, to run $1 on ${DEBILO_DIR}
    FILES_NEEDED=$(show_link_deps $1)
    # define the special destination file naming operation
    DST_FILE_OP=$(echo "echo \${EACH_FILE} | sed 's,^,${DEBILO_DIR},g'")
    for EACH_FILE in ${FILES_NEEDED}; do
        DST_FILE=$(eval "$DST_FILE_OP")
        # if $DST_FILE does not exist, copy file
        [ -f $DST_FILE ] || {
            [ -L $EACH_FILE ] && \
                SRC_FILE=$(dirname $EACH_FILE)/$(readlink $EACH_FILE) || \
                SRC_FILE=$EACH_FILE
            # directory location of $DST_FILE
            DST_DIR=$(dirname $DST_FILE)
            [ -d $DST_DIR ] || mkdir -p $DST_DIR
            cp -p $SRC_FILE $DST_FILE
        }
    done; return 0
}

```

```

# A-2) Copying necessary files
copy_files_for /bin/busybox
copy_files_for /bin/mount
copy_files_for /sbin/halt
copy_files_for /sbin/pivot_root

# A-3) Setting up necessary symlinks
cd ${DEBILO_DIR}/bin/
ln -s busybox sh
ln -s busybox chroot
ln -s busybox sed
cd ${DEBILO_DIR}/sbin/
ln -s halt poweroff
ln -s halt reboot
cd ${DEBILO_DIR}/etc/
ln -s /proc/mounts mtab
cd /

# A-4) Managing rc related things
mkdir ${DEBILO_DIR}/lib/lsb ${DEBILO_DIR}/etc/default \
    ${DEBILO_DIR}/etc/init.d ${DEBILO_DIR}/etc/rc0.d ${DEBILO_DIR}/etc/rc6.d

cp -a /etc/default/halt /etc/default/rcS ${DEBILO_DIR}/etc/default/
cp -a /lib/lsb/init-functions ${DEBILO_DIR}/lib/lsb/
cp -a /etc/init.d/rc /etc/init.d/ontmpfs /etc/init.d/halt /etc/init.d/reboot \
    ${DEBILO_DIR}/etc/init.d/
cp -a /etc/rc0.d/*halt ${DEBILO_DIR}/etc/rc0.d/
cp -a /etc/rc6.d/*reboot ${DEBILO_DIR}/etc/rc6.d/

# done with part (A)

# B) We execute the hwclock command below, to cover any failure to set the
#     system clock previously with hwclockfirst.sh.
#     (happened once to me when using a Dell laptop.)
hwclock --hctosys --localtime --noadjfile

```

```

# C) Fake mount for ${HOST_MNT} and ${DEBILO_DIR}

# check for /etc/mtab to be writable. if not, it's probably a symlink to
# /proc/mounts and we are done
[ -w /etc/mtab ] || exit 0

# fake mount for ${HOST_MNT}
mountpoint -q ${HOST_MNT} && {
    grep -E --no-messages "^[^ ]+ +${HOST_MNT} +" /proc/mounts | \
        while read HOSTFS x FSTYPE rest;
            do mount -f -t $FSTYPE $HOSTFS $HOST_MNT; done
}

# fake mount for ${DEBILO_DIR}
# our tmpfs options
TMPFS_OPTS="size=5M,mode=0755"
mount -f -t tmpfs -o $TMPFS_OPTS tmpfs ${DEBILO_DIR}

# done with part (C)

exit 0

```

```
#####
```



```

#####
# file: /etc/init.d/debilo_end
#! /bin/sh
### BEGIN INIT INFO
# Provides:          debilo_end
# Required-Start:
# Required-Stop:
# Should-Start:
# Default-Start:    0 6
# Default-Stop:
# Short-Description: Prepare ${DEBILO_DIR} for pivot_root.
# Description:
### END INIT INFO
PATH=/usr/sbin:/usr/bin:/sbin:/bin
# some checks that need to be done before we go on
# 1. the /dev/.initramfs_debilo file must exist
[ -f /dev/.initramfs_debilo ] && . /dev/.initramfs_debilo || exit 0
# 2. by sourcing /dev/.initramfs_debilo , we should have ${DEBILO_DIR} set
#    and it should be a mounted directory
[ -d "${DEBILO_DIR}" ] && mountpoint -q "${DEBILO_DIR}" || exit 0
# if we pass the tests, we assume everything is ok
. /lib/lsb/init-functions
do_stop () {
    /etc/init.d/autofs stop
    mount -n --move "${DEBILO_DIR}" /root && \
        mkdir /root/proc /root/sys /root/dev /root/old-root || exit 1
    mount -n --move /dev /root/dev
    mount -n --move /sys /root/sys
    mount -n --move /proc /root/proc
    # pivot-root op
    cd /root
    pivot_root . old-root
    exec chroot . etc/init.d/ontmpfs stop <dev/console >dev/console 2>&1
}
case "$1" in
    start)
        # No-op
        ;;
    restart|reload|force-reload)
        echo "Error: argument '$1' not supported" >&2
        exit 3
        ;;
    stop)
        do_stop
        ;;
    *)
        echo "Usage: $0 start|stop" >&2
        exit 3
        ;;
esac
#####

```

```
#####
# file: /etc/init.d/ontmpfs
#! /bin/sh
### BEGIN INIT INFO
# Provides:          ontmpfs
# Required-Start:
# Required-Stop:
# Should-Start:
# Default-Start:    0 6
# Default-Stop:
# Short-Description: umount loopback related devices before turning PC off.
# Description:
### END INIT INFO

PATH=/sbin:/bin

# source /dev/.initramfs_debilo file
. /dev/.initramfs_debilo

. /lib/lsb/init-functions

do_stop () {
    log_action_begin_msg "Unmounting loopback root filesystem"
    umount /old-root
    log_action_end_msg $?
    log_action_begin_msg "Unmounting host filesystem"
    umount ${HOST_MNT}
    log_action_end_msg $?
}

case "$1" in
    start)
        # No-op
        ;;
    restart|reload|force-reload)
        echo "Error: argument '$1' not supported" >&2
        exit 3
        ;;
    stop)
        do_stop
        ;;
    *)
        echo "Usage: $0 start|stop" >&2
        exit 3
        ;;
esac
#####
```

APPENDIX C
NEW VERSION SCRIPTS

```
#####
# file: /etc/init.d/debilo_begin
#! /bin/sh
### BEGIN INIT INFO
# Provides:          debilo_begin
# Required-Start:    hwclockfirst.sh checkfs.sh
# Required-Stop:
# Should-Start:
# Default-Start:     S
# Default-Stop:
# Short-Description: Continuing to setup debilo for further use.
# Description:       Copying necessary files to ${DEBILO_DIR}, remedy some
#                    issues regarding hwclock.
### END INIT INFO
PATH="/usr/sbin:/usr/bin:/sbin:/bin"
# some checks that need to be done before we go on
# 1. the /dev/.initramfs_debilo file must exist
[ -f /dev/.initramfs_debilo ] && . /dev/.initramfs_debilo || exit 0
# 2. by sourcing /dev/.initramfs_debilo , we should have ${DEBILO_DIR} set
#    and it should be a mounted directory
[ -d "${DEBILO_DIR}" ] && mountpoint -q "${DEBILO_DIR}" || exit 0
# if we pass the tests, we assume everything is ok
echo "Continuing to setup debilo for use .."
# Executing the necessary tasks
# A-1) Functions used to copy the necessary files
show_link_deps() {
    # shows us the files needed off of / to run $1
    echo ${1}
    /lib/ld-linux.so.2 --list ${1} | sed '/=> \\/ !d;s,^.* \(/.*\) .*$,\\1,g'
}
copy_files_for() {
    # copies the files needed, to run $1 on ${DEBILO_DIR}
    FILES_NEEDED=$(show_link_deps $1)
    # define the special destination file naming operation
    DST_FILE_OP=$(echo "echo \${EACH_FILE} | sed 's,^,${DEBILO_DIR},g'")
    for EACH_FILE in ${FILES_NEEDED}; do
        DST_FILE=$(eval "$DST_FILE_OP")
        # if $DST_FILE does not exist, copy file
        [ -f $DST_FILE ] || {
            [ -L $EACH_FILE ] && \
                SRC_FILE=$(dirname $EACH_FILE)/$(readlink $EACH_FILE) || \
                SRC_FILE=$EACH_FILE
            # directory location of $DST_FILE
            DST_DIR=$(dirname $DST_FILE)
            [ -d $DST_DIR ] || mkdir -p $DST_DIR
            cp -p $SRC_FILE $DST_FILE
        }
    done; return 0
}
}
```

```

# A-2) Copying necessary files
copy_files_for /bin/busybox
copy_files_for /bin/mount
copy_files_for /sbin/halt
copy_files_for /sbin/pivot_root
mkdir ${DEBILO_DIR}/etc/init.d/
cp -a /etc/init.d/ontmpfs ${DEBILO_DIR}/etc/init.d/
# A-3) Setting up necessary symlinks
cd ${DEBILO_DIR}/bin/
ln -s busybox sh
ln -s busybox chroot
ln -s busybox sed
cd ${DEBILO_DIR}/sbin/
ln -s halt poweroff
ln -s halt reboot
cd ${DEBILO_DIR}/etc/
ln -s /proc/mounts mtab
cd /

# A-4) Setup the /etc/rc7.d/ directory
# remove rc7.d if it exists
[ -e /etc/rc7.d ] && rm -rf /etc/rc7.d
# make /etc/rc7.d a mount from a real directory in ${DEBILO_DIR}
mkdir /etc/rc7.d ${DEBILO_DIR}/etc/rc7.d && mount -n --bind ${DEBILO_DIR}/etc/rc7.d
/etc/rc7.d
# copy /etc/rc6.d/ into /etc/rc7.d
cp -a /etc/rc6.d/* -t /etc/rc7.d/
# remove reboot script
rm -f /etc/rc7.d/*reboot
# create symlink to the several scripts, including the error inducing script
[ -x /etc/init.d/induce_err ] || exit 1
cd /etc/rc7.d/
ln -s ../init.d/udev K95udev
ln -s ../init.d/killprocs S95killprocs
ln -s ../init.d/induce_err S99induce_err
cd /

# B) We execute the hwclock command below, to cover any failure to set the
# system clock previously with hwclockfirst.sh.
# (happened once to me when using a Dell laptop.)
hwclock --hctosys --localtime --noadjfile

exit 0

```

```
#####
```

```
#####
# file: /etc/init.d/ontmpfs
#!/bin/sh
PATH=/sbin:/bin
[ $RUNLEVEL -eq 7 ] || exit 1
# source /dev/.initramfs_debilo file
. /dev/.initramfs_debilo
echo "Unmounting all loopback root filesystem related mounts"
umount /dev/.static/dev
umount -ld /old-root
echo "Unmounting host filesystem"
umount ${HOST_MNT}
[ -e /dev/.shutmedown ] && \
    { rm /dev/.shutmedown; echo 'Shutting Down.'; halt -d -f -i -p; } || \
    { echo 'Rebooting.'; reboot -d -f -i; }
#####
# file: /etc/init.d/induce_err
#!/bin/sh
echo We need this script to induce an error on our system.
echo It forces init to invoke /sbin/sulogin, which we will
echo use for the next step in our shutdown/reboot process.
echo Just enter the root password and press enter.
echo At the prompt, type: . /etc/nextstep
echo and press enter.
exit 1
#####
# file: /etc/nextstep
# if you do not know what you're doing, please do not edit this file! :)
. /dev/.initramfs_debilo
[ $RUNLEVEL -eq 7 ] && {
HALTOPT="R"
echo 'Pick an action: [S]hutdown or [R]eboot, default action is reboot.'
read -nl HALTOPT
[ $HALTOPT = "S" -o $HALTOPT = "s" ] 2> /dev/null && \
    { touch /dev/.shutmedown; echo -e '\bSystem will be shut down.'; } || \
    echo -e '\bSystem will be rebooted.'
sleep 1
cd / && mount -n --move "${DEBILO_DIR}" /root && cd /root
mkdir proc sys dev old-root
mount -n --move /proc proc
mount -n --move /sys sys
mount -n --move /dev dev
}
[ $RUNLEVEL -eq 7 ] && cd /root && pivot_root . old-root
[ $RUNLEVEL -eq 7 ] && exec chroot . etc/init.d/ontmpfs <dev/console >dev/console 2>&1
#####
```

APPENDIX D
SCRIPTS USED TO GENERATE ARGUS GRAPHS

```

#####
# file: genragraph.sh
# generate byte and packet average graphs with a split output
# for traffic originating from the inside or outside
#!/bin/sh
binsize=(ls 5m)
argusfile=(WedApr4-2007 ThuApr5-2007)
ipadd=(160.0.46.159 160.0.46.10 160.0.46.16 160.0.46.17)
HGT=480
WTH=640
for BS in {0,1}; do
  for AF in {0,1}; do
    for IA in {0,1,2,3}; do
      ragraph bytes dport -M ${binsize[$BS]} -height $HGT -width $WTH \
        -fill -r /tmp/${argusfile[$AF]} \
        -title "${binsize[$BS]} avg WWW byte traffic for ${ipadd[$IA]}" \
        -w /tmp/${argusfile[$AF]}_${ipadd[$IA]}_${binsize[$BS]}_www_bytes.png \
        - src ${ipadd[$IA]} and dst port 80

      ragraph pkts dport -M ${binsize[$BS]} -height $HGT -width $WTH \
        -fill -r /tmp/${argusfile[$AF]} \
        -title "${binsize[$BS]} avg WWW packet traffic for ${ipadd[$IA]}" \
        -w /tmp/${argusfile[$AF]}_${ipadd[$IA]}_${binsize[$BS]}_www_packets.png \
        - src ${ipadd[$IA]} and dst port 80
    done
  done
done
#####

```



```

#####
# file: genragraph_nosplit.sh
# generate byte and packet average graphs without split output
# for traffic originating from the inside or outside
#!/bin/sh
binsize=(ls 5m)
argusfile=(WedApr4-2007 ThuApr5-2007)
ipadd=(160.0.46.159 160.0.46.10 160.0.46.16 160.0.46.17)
HGT=480
WTH=640
for BS in {0,1}; do
  for AF in {0,1}; do
    for IA in {0,1,2,3}; do
      ragraph bytes dport -M ${binsize[$BS]} -height $HGT -width $WTH \
        -fill -split -r /tmp/${argusfile[$AF]} \
        -title "${binsize[$BS]} avg WWW byte traffic for ${ipadd[$IA]}" \
        -w
      /tmp/${argusfile[$AF]}_${ipadd[$IA]}_${binsize[$BS]}_www_bytes_nosplit.png \
        - src ${ipadd[$IA]} and dst port 80

      ragraph pkts dport -M ${binsize[$BS]} -height $HGT -width $WTH \
        -fill -split -r /tmp/${argusfile[$AF]} \
        -title "${binsize[$BS]} avg WWW packet traffic for ${ipadd[$IA]}" \
        -w
      /tmp/${argusfile[$AF]}_${ipadd[$IA]}_${binsize[$BS]}_www_packets_nosplit.png \
        - src ${ipadd[$IA]} and dst port 80
    done
  done
done
#####

```