Virtual Machine Technologies and Their Application In The Delivery Of ICT

William McEwan

Christchurch Polytechnic Institute of Technology
Christchurch, New Zealand

mcewanw@cpit.ac.nz

ABSTRACT

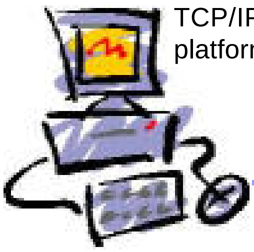
Virtual Machine (VM) technology was first implemented and developed by IBM corporation in the early 1960's as a mechanism for providing multi-user facilities in a secure mainframe computing environment. In recent years the power of personal computers has resulted in renewed interest in the technology. This paper begins by describing the development of VM. It discusses the different approaches by which a VM can be implemented, and it briefly considers the advantages and disadvantages of each approach. VM technology has proven to be extremely useful in facilitating the teaching of multiple operating systems. It offers an alternative to the traditional approaches of using complex combinations of specially prepared and configured OS images installed via the network or installed permanently on multiple partitions or on multiple physical hard drives. VM technology has proven equally useful in the practical teaching of data communications, where complex internets have to be regularly constructed and reconfigured in order to study the underlying communication protocols (e.g. TCP/IP). It is also of immense use as a platform for research into these somewhat related areas - a virtual machine or network of virtual machines can be specially configured, allowing an ordinary user supervisor rights, and it can be tested to destruction without any adverse effect on the underlying host system.

This paper hopes to also illustrate how VM configurations can greatly reduce our dependency on special purpose, complex, and expensive laboratory setups. It also suggests the important additional role that VM and VNL is likely to play in offering hands-on practical experience to students in a distance e-learning environment.

Keywords: Virtual Machines, operating systems, networks, e-learning, infrastructure, server hosting.

1. INTRODUCTION

Virtual Machine (VM) technology is not new. It was implemented on mainframe computing systems by the IBM Corporation in the early 1960's (Varian 1997 pp 3-25, Gribben 1989 p.2, Thornton 2000 p.3, Sugarman 2001 p.2). As the power of personal computers continues to grow, there has been an upsurge of interest in VM: in the academic research community, the Open Source community, and also as a commercial concern. VM technology is also being used to simulate old machines for historical reasons (Burnet and Supnik 1996, Berndtsson 1999).



A Virtual Machine is an abstraction in software of a real physical machine. It is run as a standard user application on a normal physical machine Operating System (OS). The physical machine it runs on top of is generally referred to as 'the host' (the OS of this host being referred to as 'the host OS'). The VM itself effectively emulates a physical machine and comes with its own hard drive and physical devices. The VM can therefore be installed with an OS of its own (which can often be a different OS to that of the underlying host). The OS used on the VM is referred to as the 'guest OS' (Sugarman 2001 pp 2-3, VMware WUM p.12).

The VM can generally be given access to the underlying physical devices via its own virtual devices. There are different schemes by which this virtualisation is achieved. A VM can often be provided with more virtual devices than those that are available to the underlying host system. Many VM's can usually be run on the same host system, and most implementations allow them to be provided with their own virtual network card or cards and networked together. From the users point of view such machines look just like any normal machine on the network. They can be configured, for example, to take part in TCP/IP communications or to provide network services (VMware WUM pp 239-296).

This paper focuses on the history and development of VM technology, on different techniques for achieving VM, and on its uses. In particular, the practical use of VM technology as a strategically useful methodology for implementing a Virtual Network Laboratory (VNL) infrastructure is detailed (McEwan 2001 - VNL). The VNL infrastructure is shown to be extremely useful in a local technical education delivery context but is likely to have even greater positive implications as a distance e-learning laboratory facility.

2. THE DEVELOPMENT OF VM

In 1964, IBM designed a new operating system called Control Program-40 (CP-40) for their System 360 computer. Inherent in that OS was the support for virtual memory and a fixed number (14) of 256kB RAM virtual S/360 machines: "...they would be able to run any of IBM's S/360 operating systems in a virtual machine" (Varian 1997 pp 10-12).

In 1972 IBM introduced their new VM/370 OS for the System 370. This was the combination of the control program CP and the virtual machine OS called CMS (Varian 1997 p.29). During the 1980's IBM introduced their Object Code Only (OCO) policy. Prior to that customers had always received source code with the system and were able to fix any bugs as they found them. After OCO, customers suffered long delays waiting on IBM fixes and dissatisfaction grew. "IBM... not noticing until too late that the rest of the industry (and many of its customers) were moving rapidly toward open systems" (Varian 1997 p.56). VM/370 survived and even made it into a PC: "Late in 1991... the Personal/370 card... was a real System/370 on a card... and ran as a co-processor in a PS/2... operating systems, such as the VM/ESA 370" (Varian 1997 p.64). More recently Linux has been ported by IBM to run on its VM/System 390 machine: "41,400 simultaneous Linux images before the virtual machine ran out of resources" (Thornton 2000 p.4).

3. CLASSES OF VIRTUAL MACHINE

The speed that a VM runs is determined by what proportion of its underlying software can run natively on the underlying processor and what proportion needs to be "virtualised" (i.e. emulated in software). The following classification is from Robin (2000 pp 2-4):

- "A virtual machine monitor (VMM) allows multiple operating systems to run concurrently on virtual machines (VM's) on a single hardware platform. The amount of software and hardware execution of processor instructions determines if one has a complete software interpreter machine (CSIM), hybrid VM (HVM), VMM, or a real machine...
- A real machine uses only direct execution: the processor executes every instruction of the program directly.
- A CSIM uses only software interpretation: a software program emulates every processor instruction...
- A VMM requires that a 'statistically dominant subset' of the virtual processor's instructions be executed on the real processor (Goldberg 1972).
- An HVM is a VMM that uses software interpretation on all privileged instructions... whereas a VMM may directly execute some privileged instructions."

Note that, in terms of performance, the order from slowest to fastest is: CSIM, HMM, VMM, real machine. Bochs i86 CPU emulator is a CSIM. VMware (Sugarman 2000 pp 3-12), Plex86 (Lawton 1999), and UML (Dike 2000 pp 2-7) use variations on VMM.

The Wine Project (and thus LindowsOS) instead uses Microsoft OS/API emulation (Robins 2000 p.12).

Robins (2000 pp 5-6) analysed the Intel Pentium instruction set and found that it had seventeen documented instructions that were 'sensitive' (i.e. needed to be run in OS kernel supervisory mode) but did not automatically generate a trap (software interrupt) signal. This means that there is no direct way for the VMM to know when a user-space VM issues them. The VMM needs to somehow catch these 'sensitive' instructions so that it can simulate them. The Pentium is therefore said to be non-virtualisable (the Digital Alpha on the other hand is virtualisable). Since each VM is being run as a normal user application, most of the problem of creating VM's on an Intel system therefore revolve around ways of detecting when a VM has issued one of these 'sensitive' but non-trappable instructions. . In UML, this facility is provided by a call tracing facility available in Linux called the ptrace system (Dike 2000 pp 2-7).

4. USING VM IN THE DELIVERY OF OS SUBJECT TEACHING

A few years ago, CPIT, like everyone else, were struggling with the complex administrative problems involved in teaching multiple OS systems in practical teaching laboratories. Many schemes were tried with greater or lesser success (e.g. multiple partitions, multiple, perhaps removable, hard drives, downloadable specially configured OS images) but none of these proved very satisfactory. Special facilities were required; specially constructed and configured laboratories. It was expensive, complex, a constant technical battle to administer and maintain. Then, two years ago, we started using VMware (VMware WUM). Almost immediately many of our technical problems disappeared. Suddenly we were able to use almost standard teaching laboratories where once we had multiple partitions and/or other complex schemes. We did still physically install a separate hard drive on the systems for the OS classes - but it wasn't a requirement, only a convenience. Each workstation was now only installed with one host operating system (Win NT or Win2000 or XP or Linux - whatever is preferred and whatever works with VMware). On top of that we installed the VMware application. No more special partitions (we never could decide what size to optimally make them, we never could determine how many partitions we would end up needing for the number of classes and individual operating systems we had to teach…). Now we just need to fire up a VM (i.e. run the VMware wizard and configure it with whatever virtual physical devices we want), install any OS we like onto it and build a bank of VM different guest OS images on each machine. The VM hard disk is actually stored as nothing but a normal file on the underlying host file system (VMware WUM pp 178-184). Want to get rid of a VM installation? Just delete its folder… And that is almost the end of the story when it comes to using VMware. Yes, occasionally there are some compatibility issues to sort out - between VMware and the host and guest OS configurations. For example, we often run Linux on top of VMware for teaching our UNIX classes and sometimes when a new X server GUI is released it isn't first supported with a suitable VMware driver - but these things are 'usually' sorted out fast - upgrades and patches are posted and everything becomes stable again. Sometimes it is best not to upgrade too early anyway - wait till all the bugs have been sorted out… VMware works for us, and VMware works well. No doubt there is an effect on performance on the guest OS but we hardly notice any. And of course there is a minimum hardware requirement, the more memory and the faster the CPU the better (VMware WUM pp 13-14).

We are not restricted to running only one VMware machine at a time. In data comms network classes we regularly get the students to fire up three or four VM machines on their local workstation, configure routing on them, and use them in experimental configurations (VMware WUM pp 240-296). Students can build mini-internets, completely sandboxed from the main campus. Individual VM machines have been set up as servers (name servers, web servers whatever). Suddenly we don't need lots of physical equipment and bits of hardware and cables everywhere. It isn't perfect - virtualization adds complexity to the overalls system load - it makes often heavy use of the system resources - crashes can occur (though VMware has proven to be remarkably resilient). Once we made a serious mistake and one

VMware machine running on a student desktop was wrongly configured and became an unwanted accidental DHCP server on the main CPIT campus network. Our campus network support division were far from happy - but we learned from that experience. Since then we know how to use VMware better. Network support IT division were able to help - using VLAN capability on the campus core routing switches they were able to isolate certain VMware-using labs onto their own VLANS. Generally, IT division support needs have fallen to almost nothing in these specialised areas - we look after the VMware parts of things ourselves. Yes, that is a burden on teaching staff - but the amount of time we spend on installation and configuration is actually much reduced to what it was before we used VMware.

5. USING USER-MODE LINUX IN OUR VIRTUAL NETWORK LABORATORY

VMware is great, but it is resource intensive. Server versions are available (VMware GSX/ESX) but they require heavy duty, high performance servers and don't easily or cheaply scale sufficiently for our needs - the GSX allows 2-8 simultaneous VM's the ESX 4-20 VM's. The Open Source User-Mode-Linux (UML or uml) project (Dike 2000, 2001, UML) helps fill the gap. The author of this paper has been actively involved with UML configuration testing for about 18 months now (McEwan 2001 - VNL). UML is in its infancy, in a state of current development, but it has generally proved to be very stable. At CPIT we have used UML to construct a network of Linux based uml VM servers all running on the one host Linux system. It has been running continuously since October 2001 and is regularly accessed by students. This demonstrates its reliability.

Details of the design of our VNL have been published at the official Open Source web site for uml (McEwan 2001 - VNL). Many of the details of the final configuration shown there were ironed out, following long email discussions, with the lead developer of UML [Dike, Personal communication] and by closely following developments on the associated users and developers mailing lists. At CPIT, we have created various bash shell scripts to automate the creation of our VNL. These are freely available in a UNIX tar file from the official UML web-site (McEwan 2001 - VNL). As can be seen from Figure 1, one of the uml machines in our VNL is configured as a routing firewall. It routes traffic from the other uml machines, via the host machine, onto our main campus network. Using it as a routing firewall allows us to add extra security to our VNL (e.g. students can be allowed by the firewall to ssh into the system but not to ssh out).

Work continues on this development and we hope to have an updated version of our work published later this year at the official UML web-site. In practice our VNL uses an official class C address range (i.e. our uml machines are real Internet machines, we are not using Network Address Translation or proxies to communicate with them - though we could…). The semi-automating network build scripts referred to above can be given any class C address range as a build parameter.

Currently, the maximum number of uml VM's we have had running simultaneously on the single host server is fifty. This on a Pentium III/1 GHz machine with 384 MByte of RAM and 512 MB of swap. (Compare this with the typical VMware GSX/ESX server version figures previously mentioned). UML is very resource friendly. On an old Pentium 100/ RAM:32MB/HD:500 MB/swap:64MB machine we have had ten uml machines up and running networked! - albeit slowly…lots of disk swapping activity... For performance reasons, we aim to have twenty uml instances (enough for one class of students) available on our remotely accessible VNL server. When we need to expand on that we shall simply install another server and join it to the network. The VNL is part of the physical campus network - the student doesn't know if she is working on an actual machine or on a virtual machine - it is totally transparent. Each of these VM uml machines is self-contained, has its own IP address; secure shell, ftp and telnet servers.

Students can be assigned their own uml machine on this server and given supervisor (root) user rights (yet remain as normal users only on the host machine). The benefits are obvious (Dike 2001 pp 10-11). A student can now sit at any computer in any lab on campus and login remotely to the Virtual Network Lab of uml machines. If they have a local X server (Wong et al. 2000, X.org) they can run X based applications remotely on their uml machine, with a familiar X window environment, controlling and seeing everything from their local workstation. Alternatively we sometimes use a VNC client/server arrangement for remote control (AT&T). More often, we use Java

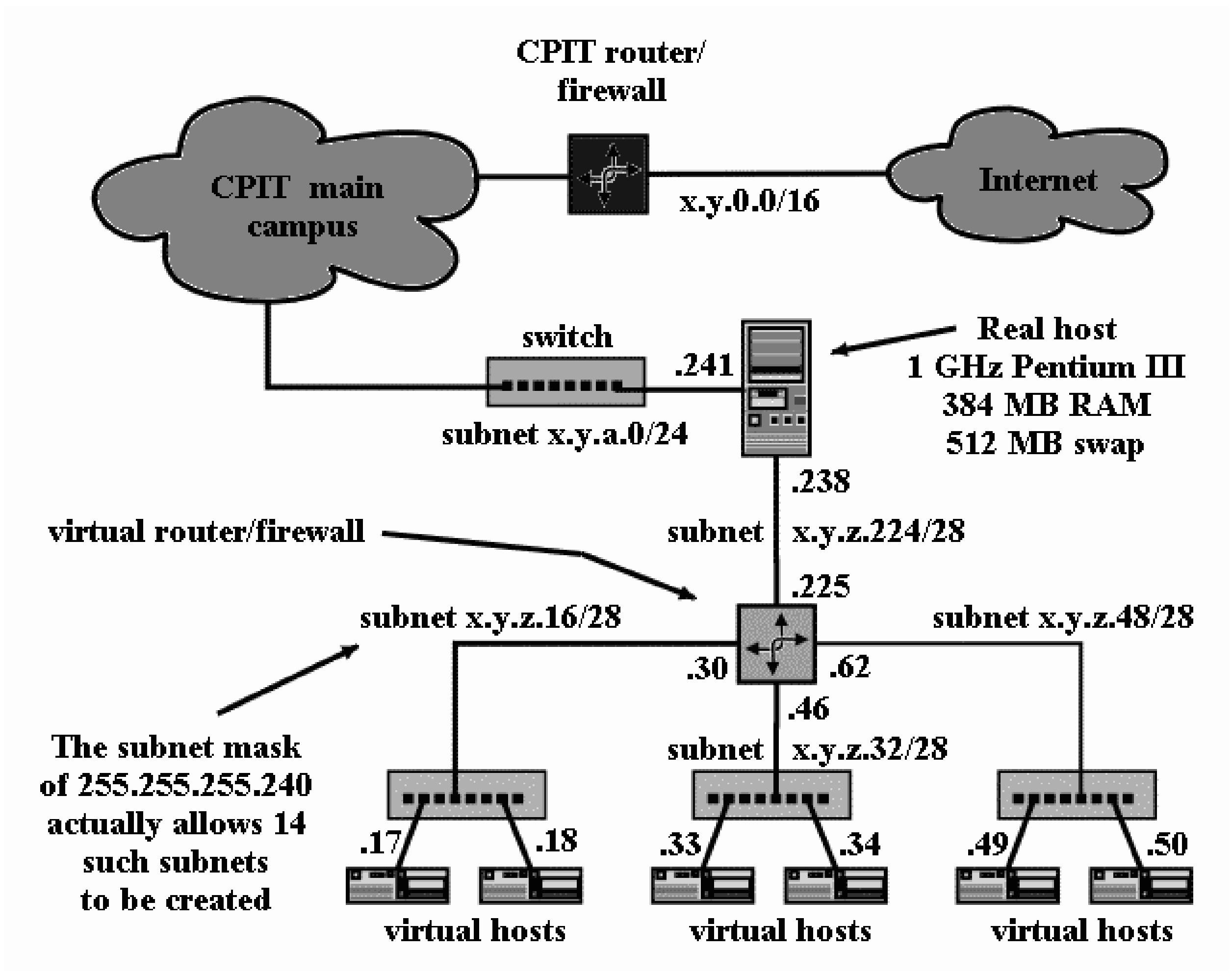


Figure 1. The CPIT Virtual Network Lab configuration and its connection to the main campus network.

applets to deliver a secure shell client (AppGate) and a simple Java X server (JCraft) to campus workstations - i.e. the Java applets are delivered remotely from a web server. We also use Cygwin for a zero cost X server solution on some of our lab installations (Cygwin/XFree86).

We plan to add some VMware VM's to this uml VNL running on the host Linux server. That will allow us to install Microsoft based OS on guest VMware machines as part of a mixed OS VNL. (Students can already install uml based network configurations on top of their local workstation VMware setups running on Windows NT or 2000 or XP, if they so choose - there is endless flexibility and scope).

The uml VM machines are in regular use by students - we can, and sometimes do, deliver a whole practical lab course on OS or data comms using the VNL server machine remotely from any standard computing lab.

6. SOME NEAR FUTURE DEVELOPMENTS

Currently the VNL server is inside the campus firewall. Once we have finished on-campus testing it is intended to make it accessible from outside. That will open up many new possibilities. We shall be able to use the VNL for distance e-learning practical lab sessions - perhaps in conjunction with Blackboard for interactive chat and whiteboard etc. Owing to a general interest in XML and Python within the School of Computing, an interesting alternative to Blackboard might possibly be contructed using the Open Source: Zope Content Management Framework (Digital Creations) and Jabber XML based chat/conferencing server, or something similar. The VNL is an excellent environment for experimenting with configurations such as those. Different uml machines can be employed for different services - one uml per server. This allows delegation of admin responsibilities and simple per-server configuration. The VNL host machine is currently also set up as a Kylix server but once again a better strategy might be to install Kylix on its own uml virtual machine inside the VNL.

7. SOME ALTERNATIVE VM IMPLEMENTATIONS

A varied collection of VM implementations is listed in the references (UML, VMware, Connectix, Plex86, Bochs, a386, The Brown Simulator, Ashton:Solaris MINIX, The Wine Project, LindowsOS, Win4Lin, z/VM, Mac-on-Linux, Disco, SimOS, Virtutech Simics). The open source projects: Plex86, Bochs, and the Wine Project are of particular interest.

Bochs (Lawton 1999), is a complete emulation of an x86 CPU and comes with an implemented BIOS. It has been ported to several operating systems (including Win95, NT and Linux) and various platforms (including the MAC 68x and Amiga PPC). At CPIT we have successfully run Bochs inside Win95 (and booted up a small Linux distribution on it). Unfortunately, since it emulates every x86 instruction in software it is very slow. Recent versions of Bochs can be attached to a network. It can also be accessed on a network via VNC (AT&T Labs).

Plex86 (Lawton 1999,2000) is similar in concept to VMware, but it is still in development and unfortunately work on it seem to have slowed of late.

Wine is an implementation of the Microsoft Windows 3.x and Win32 APIs on top of X and Unix. It can run a surprising number of modern Windows binary applications (e.g. Office), albeit with some bugs. It can only run on x86 platforms however. It has been in development for many years owing to the complex ever changing nature of the Windows APIs. Note that LindowsOS is a commercial product that uses Wine code as its base. LindowsOS is much in the news recently owing to a legal battle with Microsoft over its name (Berger 2002).

8. CONCLUSION

Based on the facilities offered by VM technology it has become possible to build mixed Virtual and Physical Network Laboratory facilities (VPNL). E-learning is becoming of increasingly great importance in the education market. It is hoped that this paper has illustrated the strategic importance that VM technology will have in that market place. It can provide an infrastructure that allows us to offer student-centred hands-on practical course delivery via remote technologies.

REFERENCES

a386. (A C programming library which provides a virtual machine). Accessed March 14, 2002. <http://a386.nocrew.org>

AppGate. MindTerm ssh java client applet. Accessed March 14, 2002. < h t t p : / / www.appgate.com/default_.asp>

Ashton, P., Solaris MINIX. University of Canterbury, NZ. Accessed March 14, 2002.<http://www.cosc.canterbury.ac.nz/~paul/smx.html>

AT&T Labs, VNC (Virtual Network Computing), Cambridge, UK. Accessed March 14, 2002.< http://www.uk.research.att.com/vnc/>

Berger, M., (2002). “Lindows takes on Microsoft”. InfoWorld Media Group Inc.Accessed March 14, 2002. <http://www.infoworld.com/articles/hn/xml/02/02/19/020219hnlindows.xml>

Berndtsson, T. (1999). The OSIS Project. Atari TOS, Mint, AES, and VDI emulation for Linux/m68k. Accessed March 14, 2002. < http://osis.nocrew.org/

Blackboard Inc. Accessed March 14, 2002. < http://www.blackboard.com/>

Bochs IA-32 Emulator Project. Accessed March 14, 2002. < http://bochs.sourceforge.net/>

Borland. Kylix 2 for Linux. Accessed March 14, 2002. < http://www.borland.com/kylix/>

Burnet, M., Supnik, B., (1996). “Preserving Computing’s Past: Restoration and Simulation”. The Digital Technical Journal. 8(3). Accessed March 14, 2002. The Computer History Simulation Project: <http://simh.trailing-edge.com/>

Connectix Virtual PC For Windows, Accessed March 14, 2002. < http://www.connectix.com/>

Cygwin/XFree86. Accessed March 14, 2002. < http://cygwin.com/xfree/>

Digital Creations, Zope CMF Dogbowl. Accessed March 14, 2002. < http://cmf.zope.org/

Dike, J. (2001). "User-mode Linux". Proceedings of the 5th Annual Linux Showcase & Conference, Atlanta, Georgia, USA.

Dike, J. (2000). "A user-mode port of the Linux kernel". Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, Georgia, USA.

Disco. Accessed March 14, 2002. < http://www-flash.stanford.edu/Disco/>

Fairhurst, G., Spracklen, C.T., Samaraweera, N., McEwan, W. (1993). "AN ANALYSIS OF TCP/IP IN THE CODE NETWORK". IEE Third European Conf. Sat. Communic., ECSC-3, Manchester, UK: 16-20.

Ford, B., Hibler, M., Lepreau, J., Tullmann, P. (1996). "Microkernels Meet Recursive Virtual Machines". Proceedings of the USENIX 2nd Symposium on Operating Systems Design and Implementation (OSDI '96), Seattle, Washington, USA.

Goldberg, R.(1972). "Architectural Principles for Virtual Computer Systems". Ph.D. thesis, Harvard University, Cambridge, MA, USA. Cited in Robin (2000).

Gribben.P. J. (1989). "Development of 360/370 Architecture: A Plain Man's View". EDS, UK. Accessed March 14, 2002. <http://pucc.princeton.edu/~melinda/ > downloadable: gribbin370.pdf

Jabber. Accessed March 14, 2002. < http://www.jabber.org/>

JCraft Inc., weirdx. Accessed March 14, 2002. < http://www.jcraft.com/weirdx/index.html>

Lawton, K., (2000). "plex86: an i80x86 virtual machine". USENIX Association. Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, Georgia, USA.

Lawton, K., (1999). "Running multiple operating systems concurrently on an IA32 PC using virtualization techniques". Accessed March 14, 2002. < http://www.plex86.org/research.phtml >

LindowsOS. Accessed March 14, 2002. < http://www.lindows.com/>

Mac-on-Linux. (Mac-on-Linux lets you run MacOS under Linux/ppc). Accessed March 14, 2002. < http://www.maconlinux.org/>

McEwan, W. (2001). "Virtual Network Laboratory - A Detailed Case Study HowTo". Accessed March 14, 2002. < http://user-mode-linux.sourceforge.net/case-studies.html>

McEwan, W. (2001). "Using Academic Research Methodologies to Improve the Quality of Teaching: A Case Study". In Proc. Fourteenth Annual Conference of the NACCQ, Napier, New Zealand: 83-93.

McEwan, W. (2000). "A general methodology for Link Quality evaluation and its practical application to the CODE VSAT satellite link". A thesis presented for the award of MSc Engineering (by research), University of Aberdeen, UK. (Copy kept at Christchurch Polytechnic Institute of Technology library).

Plex 86: open source PC virtualization software. Accessed March 14, 2002. < http://www.plex86.org/

Robin, J.S., Irvine, C.E. (2000). "Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor". USENIX Association. Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, USA.

SimOS. Accessed March 14, 2002. < http://simos.stanford.edu/>

Sugerman, J., Venkitachalam, G. and Lim, B-H (2001). "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor". USENIX Association. Proceedings of the 2001 USENIX Annual Technical Conference, Boston, Massachusetts, USA.

The Brown Simulator. Accessed March 14, 2002. < http://www.cs.brown.edu/software/brownsim/>

The Wine Project. Accessed March 14, 2002. < http://www.winehq.com/>

Thornton, A. (2000). "Linux on the System/390". USENIX Association. Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, Georgia, USA.

UML. "The User-mode Linux Kernel Home Page". Accessed March 14, 2002.< http://user-mode-linux.sourceforge.net/index.html>

Varian, M. (1997). "VM and the VM Community: Past, Present, and Future". SHARE 89 Sessions 9059-9061, Princeton University, NJ, USA.

Varian, M. (1983). "What Mother Never Told You About VM Service". Princeton University Computing Centre, Princeton University, NJ, USA. Accessed March 14, 2002. http://pucc.princeton.edu/~melinda/> downloadable: tutorial.pdf

VirtuTech Simics. Accessed March 14, 2002. < http://www.simics.com/>

VMware Workstation User's Manual. Accessed March 14, 2002. <http://www.vmware.com>

VMware GSX/ESX. Accessed March 14, 2002. <http://www.vmware.com>

Win4Lin., NeTraverse Inc. Accessed March 14, 2002. < http://www.win4lin.com/>

Wong, A.Y., Seltzer, M. (2000). "Operating System Support for Multi-User, Remote, Graphical Interaction". Proceedings of 2000 USENIX Annual Technical Conference, San Diego, California, USA.

X.org. Accessed March 14, 2002. < http://www.x.org/

z/VM: VM/ESA for IBM S/390 servers. Accessed March 14, 2002. < http://www.vm.ibm.com>