

# Virtual Square

Renzo Davoli

Department of Computer Science

University of Bologna, Italy

renzo@cs.unibo.it

**Abstract** — It is common to call “virtual world” the set of abstractions given to humans by computers and networks. When the computers and the networks are emulated by programs, i.e. are virtual machines and virtual networks, there is one more layer of virtuality: a kind of virtual squared world. Virtual square ( $V^2$ ) is the project to study how to exploit the possibilities given by multi layered virtuality.  $V^2$  integrates the existing technologies in system emulation with a new overlay network that is able to integrate virtual machines, real computers and processes. The virtual networking infrastructure, the educational virtual machine uMPS and the user level network virtual machine Ale4NET are original contributions made by this project. All the code developed for Virtual Square as well as all the code of the other virtual machines presented here have been released as Open Source or Free Software.

## I. INTRODUCTION

Virtual square [2] is a completely new perspective to computing and communicating. Hardware computers and networks are just physical entities where virtual computers and networks can run. What appears in terms of computers, hardware architecture, network topology, geographical location can be completely different from the real physical structure. A virtual computer is a program able to emulate in some manner a real computer, in the same manner a set of programs can implement the abstraction of a virtual network: it appears to users as a standard network but it does not exist in the reality, in the (single) virtual world.

$V^2$  system can be isolated from the rest of the world or it is possible to set up interconnection points between virtual square networks and standard networks, maybe the Internet. In this way virtual computers and networks appear completely integrated with real computers and networks. It is not possible to distinguish the ones from the others when accessing to services.

$V^2$  is a Swiss knife able to solve several problem related to compatibility, security and testing. Virtual machines can provide an emulated hardware architecture compatible with the desiderata of an application. The mismatch between real hardware and application requirements in terms of processor, peripherals or operating systems becomes just a matter of reduced performance. The possibility to design completely isolated network as well as virtual network of any topology makes  $V^2$  a versatile tool for security. The same versatility in terms of networking together with the wide range of available virtual machines are the key features for creating virtual laboratories: a workbench for a wide range of experiments.

## II. APPLICATIONS OF VIRTUAL SQUARE

### A. Virtual Square and Mobility/Migration.

Host computers and networks are only computational and communication resources for  $V^2$ .

This work was partially supported by the WebMinds FIRB project of the Italian Ministry of University, Research and Education. Verbatim copying and distribution of this entire article is permitted in any medium, provided that this notice is preserved.

It is natural for an environment like that to cope with mobility: changing a path in the real network system (e.g. changing real IP address, provider, connection technology, etc) is like to change a cable between two switches in the virtual square network. The packets get rerouted on the new path in the real network but no changes in connectivity or address take place in the virtual square world. Open connections are unaffected, just the performance is related to the changes in the underlying “virtual once” layer. In the same way a virtual machine can be stopped on one real computer, the memory and disk state saved and transferred on another computer then restarted. If the virtual machine is connected to the same virtual network on both computers the network communications can be unaffected by the change.

### B. Virtual Square and Security/Privacy.

Security is not just a matter of protecting data on storage and on communication channels. We are spreading information about ourself in the world as Hansel left pebbles along the way.

Communication channels can be encrypted and the idea of virtual machine itself has been used in many ways to create a cage for protecting data (like the sand box concept of the Java Virtual Machine). Using  $V^2$  we can be protected also by geographical localization (a traceroute in the virtual square world is completely unrelated to geographical or network topology consideration). We can also protect our pattern of traffic by changing dynamically the address and the peer to used to enter the virtual square network. It is possible to have different machines or just personalities on a system communicating using different channels and providing different data. Sensible data is sent through safe, encrypted, not locatable channels while anonymous interactions can be transmitted in clear using local interfaces. Nothing is impossible for security threats, but it is very harder for a cracker to access sensible data if he/she has not only to break the access to a machine but also the (virtual) machine cannot access the sensible data that are stored elsewhere. Two layers should be violated instead of one.

### C. Management of Hardware/Software Upgrades and Heterogeneity

The choice of an hardware/software infrastructure can be an inertia factor for evolution. The common example is the amount of investments in software that can prevent the move towards a modern and effective operating system. Using virtual square techniques it is possible to run the old software in the new environment by defining a virtual machine able to appear as the old one both in terms of hardware architecture and operating system. This solution helps in making a gracefully migration between different architecture.

#### D. Computer Science Education/Research.

A  $V^2$  can be both a medium and an object of experimentation. It is possible to install and test several of Operating Systems on different hardware topologies. This is a configuration for teaching or testing operating systems installation or administration. The FreeOsZoo project [21] provide a number of disk images with open source operating systems ready for testing. Virtual networks can be configured as private networks where students can test services, operating systems, without any concern about the danger for the security of real networks. Students can run their web services, test any kind of virtual machines, run network attacks to verify the reliability of an operating system. Everything appears as real, is completely real world consistent, but it is virtual. The confinement on the experimental space can be total (no exchange between real and virtual networks) or defined a priori with any kind of restrictions. Students can also be allowed to join this “realistic but virtual world” from home. It is sufficient to set up a tunnel between the virtual network running on their own PC and the lab virtual network at the University. Virtual Square Networks can also be used to study networking. Students can set up as many virtual networks as they need and they can implement their own topology by using switches, routers, firewalls, etc. Managing software instead of hardware makes the solution cost effective, less prone to obsolescence, and safe. All students’ virtual networks cannot interact directly with real networks and there is no need to give either student administration access on computers or permission to track real network traffic.

### III. $V^2$ MACHINES

$V^2$  supports many different virtual machines. This section presents a summary of the supported machines.

User-Mode Linux [5], [6] It is a project that realizes a complete system virtualization through system trapping. It has been released as a set of patches for the Linux kernel that defines a new virtual “um” hardware architecture. A kernel for the “um” architecture is just an executable for the host computer that include the I-O virtualization routines as well as the kernel itself. It runs at user-level. It does not require a specific kernel support in the host machine but there is patch named skas mode for the 2.4 version of Linux to increase U-ML security and performance (to reduce the number of threads and to keep the addressing space of the emulated kernel inaccessible by the emulated tasks). The support for skas mode should be included in vanilla Linux 2.6.

Qemu [1] Quoting its author’s web site: Qemu is a FAST! processor emulation using dynamic translation to achieve good emulation speed. Qemu is able to run just as a processor or as a Complete System Virtualizer. Running different executables it is possible to run single executables compiled for different processor architectures in a Linux environment or it is possible to start a virtual machine and boot an entire operating system. It runs on a number of different hardware architectures, it is currently able to run i386,ppc,arm and sparc executables and provides virtual machines emulating i386 and ppc based architec-

tures. This project is very active: new ports and features are announced on daily base. It runs completely at user-level and virtualizes completely the processor architecture.

Bochs [17] Bochs is an historical virtual machine. It runs on several host architectures (supported OS: Linux, MacOS 9/X, Windows) where it is able to create complete system virtualization of an i386 architecture. It relies on standard emulation techniques thus it is quite slow when compared to modern virtual machines. It runs completely at user-level and virtualizes completely the processor architecture.

PearPC [10] It is conceptually similar to bochs but it implements a PPC architecture instead of an i386 PC. Thus it creates a complete system virtualized PPC box able to run several Oses including Linux and MacOX 9/X. It runs on several architectures but there are special performance optimizations tailored for i386 host machines. (The author says that it runs 40 times slower than the real host performance on a i386 and 500 times slower on other CPUs, but these absolute figures are not related to any well-known benchmark). For the color taxonomy it runs at user-level achieving a completely processor virtualization.

MPS [19] MPS and uMPS (micro MPS) have been designed for educational purposes. Like Qemu, Bochs and PearPC, MPS is a complete virtual system of an Mips based computer (user-level, complete processor virtualization). It is a workbench for computer science students to run their experimental operating systems in a real-world consistent virtual computer while stripping off unnecessary complexities. It is mentioned here as uMPS comes (soon) with a virtual Ethernet interface that can join the Virtual square world.

Ale4NET Application Level Environment for Networking (Ale4NET) has just been released in alpha version. It is an I-O Virtualization only system: with Ale4NET unix processes (or group of processes) can join a virtual network support. There is neither processor nor system emulation, just the network calls are diverted to a Ale4NET daemon that gives a completely different perspective of the connectivity. Ale4NET can be used as a bridge to enter the virtual square world from the host machine at user-level. In fact, differently from using tun-tap based solutions, Ale4NET virtualizes the network instead of creating an OS accessible interface, thus there is no need for root access. Ale4NET uses system call trapping through the dynamic library preloading technique: libc interface routines to access system calls are overridden by Ale4NET functions that trap network accesses. Ale4NET is IPv4 and IPv6 compatible, it includes a single hybrid stack able to manage both families of protocols.

PearPC This emulator is able to provide a virtual machine based on powerpc processor. There is a (quite fast) CPU emulator for the IA32 architecture hosts and a slower standard CPU emulator for other host computers. Linuxppc and Darwin based operating systems can run on PearPC. It runs as a standard user process.

Mac-on-Linux It is a complete system virtualization for an Apple Macintosh architecture for Linux/ppc. It is able to run Linux and MacOS 9 and X. It uses system trapping

but requires a Kernel module as it relies on direct hardware access. It is very fast and effective.

#### IV. $V^2$ NETWORKS

$V^2$  networks are based on the idea to implement distributed layer 2 (Data Link) Networks. The core tool of the current implementation is in fact the Virtual Distributed Ethernet (or VDE):

- it is *Virtual* because all the basic appliances of the network are emulated by programs;
- it is *Distributed*: the communicating peers can be geographically distributed;
- it is an *Ethernet* as all the frames on the virtual network are Ethernet standard compliant.

A VDE is able to interconnect different entities:

- $V^2$  machines (see section III)
- physical computers: a VDE appears as one further network interface
- single applications or groups of applications (ale4net)

It is thus possible to create a mix of real computers, virtual machines and even processes able to communicate as they were running on different computers interconnected by a single LAN, regardless of where each computer, VM or application is running. While it may appear not realistic to use a IPv4 address for each process, IPv6 provides a very large address space where the basic ideas of networking, like network layer addressing, can be redesigned from scratch.

There are several benefits in creating layer 2 overlay networks instead of IP overlay networks:

- Virtual Networks can be interfaced to the Internet as they were ordinary LANs. It is also possible to create bridges between real LANs and virtual LANs.
- All the network protocols running on a LAN can be routed inside the virtual networks.
- There is no need for specific drivers in the end node computer operating systems.
- All the features of a real networking system can be extended to the virtual distributed LAN including: *Switching* to optimize traffic, *Minimum Spanning Tree* for fault tolerance, *Trunking* to increase the aggregate bandwidth, *Tagged packets* to support several VLANs on the same cable, etc.

#### V. VIRTUAL DISTRIBUTED ETHERNET (VDE)

A VDE is an Ethernet compliant network. Like most real Ethernet networks today, it is composed by switches (or hubs) and cables. Each computer interface is connected to a switch (or hub) port. In the same manner VDE has virtual switches (that can operate also as hubs if needed by setting a parameter) and virtual cables. A virtual switch is a program that is able to interface virtual machines, real computers or other virtual switches through virtual cables. It is the virtual counterpart of an Ethernet hardware switch or bridge: the mapping between MAC (hardware) address and port is stored in a switching table which is kept updated by inspecting the packet headers flowing through the switch. Packets for unknown MAC addresses are processed as broadcast packets and resent on all ports. Aging and timeouts are used to remove obsolete data from the switching table.

- **VDE-switch to real computer connection.** The VDE network appears to the operating system as a hardware Ethernet interface. Clearly all the virtual or real units connected to the same VDE are accessible as they were on a local network regardless of their geographical distance, but it is also possible to use bridging, filtering and routing software to interoperate between different VDE networks or between VDE networks and real networks. Administration permissions to the real computer are needed to configure the VDE interface.
- **VDE-switch to virtual machine connection.** Positive effects of using VDE for virtual machine communications include:
  1. *Use of protocols not supported by the hosting computer.* VDE networks do not use real networks but for tunneling packets between switches, thus protocols running inside the virtual networks are completely independent from the protocols used by the real computer.
  2. *Extension of the sandbox effect of virtual machines to virtual networks.* Virtual machines can be used to confine [16] the effects of computations: the file systems and devices of the hosting computer are not directly accessible from a program running on a virtual machine. The effects of an erroneous or malicious computation are thus limited to the virtual machine resources. The same idea can be applied to networks by using VDE.
- **Switch-to-Switch intercoaght.** Switches can be interconnected by cables in the virtual world exactly in the same manner real hardware switches are connected. A virtual cable is a software structure that interconnects two switches like a crossed twisted pair cable does for real switches. Packets can be forwarded between switches using standard tunneling techniques either in clear or in encrypted form.

#### VI. VDE: CURRENT IMPLEMENTATION.

Everything here discussed has been implemented as free software under the GNU public licence and is available for downloading and testing from Sourceforge [4]. The software has been designed for the GNU-Linux operating system but could be easily ported to other platforms.

VDE includes several tools:

- **Vde-switch.** It implements the switching facility. It needs root access (administration privileges) only when the `-tap` option is specified. In this latter case, in fact, the virtual network can be accessed by a virtual interface from the hosting operating system. In all other cases, when there is no local connection (on the local computer) between real and virtual connectivity the switch runs with standard user permissions.
- **Connectivity tools for virtual machines.** Several kind of virtual machines can be connected to VDE:
  - User-mode-Linux: VDE is compatible with the *daemon* protocol thus can be used instead of the *uml-switch*.
  - Qemu [1]: the tool *vdeq* provided with VDE is a wrapper to interface qemu with VDE.
  - Bochs: a VDE interface comes with its standard distribution.
- **libvdetap.** It is a tuntap emulation feature able to interface any program using the Linux kernel virtual network mod-

ule. As an example the tuntap emulation can be used to support PearPC.

- **Vdeslirp.** Slirp was a tool by Danny Gasparovski dated back to 1995. At that time Internet providers proposed two different kinds of contracts: a cheap remote terminal connection and an expensive ppp/slip service. Slirp was able to convert a terminal line into a ppp/slip access for client applications. Vdeslirp has been derived from Gasparovski's work and is able to connect an entire VDE based network to the Internet. Vdeslirp runs completely at User-level: whenever a client application tries to open a new network connection, vdeslirp catches the *connect* request. Vdeslirp does the **connect** on the real Internet for the internal application and then forwards all the packets from/to the requesting internal application. From the Internet point of view (and from the host computer operating system) it is like all the connections were generated by the vdeslirp program itself. It supports TCP on IPv4. The IPv6 version is already under development. The VDE slirp tool has also an embedded DHCP server that can be activated as an option so that a VDE network with vdeslirp behaves exactly as it were an IPv4 masqueraded network with DHCP auto-configuration for client machines.

## VII. APPLICATION LAYER ENVIRONMENT FOR NETWORKING (ALE4NET).

Ale4net can be seen as a virtual networking tool, a virtual machine limited to networking. Ale4net overpasses the idea of network address (IP address) related to a computer or to a single interface or even to a user (uln). There is the freedom to assign IP addresses to each running process or to a group of programs.

Ale4net does not require any administration privilege (root access) to run. When ale4net is running there are no apparent functional difference: each user can run any program on the host machine using the host machine operating system. The only notable difference is that all the networking is done through virtual interfaces that the user can configure by him/herself.

Ale4net is a client server application (both client and server run with standard user privileges). When an application program makes a library or system call for networking, that call is caught and rerouted to the server where there is an entire user-space network stack able to create the abstraction of virtual interfaces. Ale4net servers can be (and generally are) connected to VDE switches. The client server approach gives to the user the freedom to choose how many different networking interfaces or environments he/she needs. In fact, clients connected to the same server share the same view of the network, i.e. the same virtual interfaces with the same addresses, routing etc.

## VIII. ALE4NET PROTOTYPE

A prototype for Ale4net can be downloaded for testing from sourceforge[3].

The current implementation runs on the GNU-Linux operating system and uses the dynamic library method override to catch the networking calls. It inherits some of the client-server code from the Alpine project [9], [?]. The Ale4Net

server includes an hybrid IPv4-IPv6 stack named lwip-v6, also realized within this research work, that has been designed on the basis of the light weight IP stack [8], [7]. It is an hybrid stack as it is a single stack IPv6 with specific code at the IP layer to handle IPv4 packets. TCP, UDP code is completely shared. Lwip-v6 provides a Berkeley socket interface where each call to IPv4 address is converted to IPv6 embedded IPv4 address (like 0::ffff:127.0.0.1, defined in RFC 1884 [13]). Thus lwip-v6 behaves as a dual stack.

In Ale4net this lwipv6 stack is used by the server to handle the network traffic for the clients.

When an Ale4Net server is running, a user can join the virtual network by

- setting the LD\_PRELOAD environment variable to configure the GNU-Linux dynamic linker/loader to run the Ale4Net client library and
- setting the environment variable to choose the specific Ale4Net server for networking.

The shell returns the standard prompt and everything can be done. The user gets also the right to run administrative commands for networking configuration (like *ifconfig* and *ip*). In fact these commands, which are the standard commands provided with the Linux distribution, have their calls rereouted to the ale4net server and operate on the user defined networking environment. The user can define the IP (v4 and v6) addresses, the routing, MTU and all the communication parameters relative to the specified Ale4net server.

## IX. RELATED WORK

Several tools do exist that are able do cover part of the features of VDE. Point to Point Tunneling Protocols (PPTP) [12] and Level 2 Tunneling Protocols (L2TP) [23] are both data-link tunneling protocols. Both emulate a point-to-point connection able to run PPP. PPTP has internal security features while L2TP needs IPsec to create secure channels. These protocols and tools have not been designed for distributed networks but just as point to point connections for VPN. There are many examples of tool for IP tunneling like openVPN [25], Zebedee [26], IPsec [14] e.g. the Free SWAN [11] implementation. These tools are tailored to IP and thus less general with respect to VDE. Virtual Tunnel (VTUN) [15] and Virtual Private Ethernet (VPE also named GNU VPE or GVPE) [18] have more similarities with VDE. In fact VTUN and VPE can create data link Ethernet compliant tunnels. All the tools here listed are interfaced to the tuntap kernel driver thus always need to have root access for their installation. Moreover they have been designed for real operating systems and does not interface virtual machines. VDE networks are also similar to Honeynets proposed in [22]. Both VDE and Honeynets are able to interconnect virtual machines and real networks. Honeynets however are designed for security and run on a single computer, VDE provides a general purpose distributed environment.

Ale4net is an evolution of Alpine [9] and Alpine4linux [20] ideas. Alpine is tailored to FreeBSD, Alpine4Linux is another independent implementation for Linux. Ale4net inherits from Alpine the idea of client server support for user processes through dynamic library method overriding. Anyway both projects are intended just for local use on a single

computer, not as support for a distributed virtual networking, need root access to start the server and IPv6 is unsupported.

Emulab [24] is similar to virtual square for the distributed approach to network emulation and for configurability. The main goal of emulab, however, is to act as a single virtual machine for distributed system simulation. Emulab is a sort of huge machine where researchers can test distributed systems. Virtual Square is a general purpose tools that can be used by a single user, say a student or a researcher, as a network tunnel or to emulate a distributed system on his/her own laptop or for several other application. The main goal of emulab is an effective emulation of large systems within a research laboratory. Virtual Square has been designed for personal computers, laptops, workstations and for standard day by day applications.

## X. SOME TESTS ON PERFORMANCE

This section has included just to show the effectiveness of the method, not to prove the optimality of its performances. It is clear that this general purpose solution can be less efficient than other approaches implementing tunneling or encryption or mobility separately. The choice for a virtual square solution is generally driven by its flexibility to solve a wide range of different communication, computation and security problem, by its ability to adapt to a wide range of applicative scenarios.

Table I shows the result of some tests made on VDE. The testing environment is composed by two computers connected on a switched 100Mbps Ethernet. For the sake of information completeness the configuration of the two computers is:

- PowerPC G4 1Ghz, 1GB RAM, SunGEM 10/100/1000 Ethernet Interface, running GNU-Linux kernel 2.6.3, Debian SID distribution;
- Intel G4 1Ghz, 1GB RAM, RealTeK 8169 Ethernet 10/100/1000 controller, running GNU-Linux kernel 2.6.5, Debian SID distribution.

VDE has been set up to implement a tunnel. The columns are the performance results obtained outside the tunnel (no VDE) using a unencrypted VDE cable (based on nmap) on IPv4 and using a cable with encryption (ssh). In the last two columns the same experiments have been repeated using nmap and ssh on IPv6 instead of IPv4 as a transport.

The first experiment is a standard ping (v4 and v6). Groups of twenty packets have been monitored to estimate performances. The resulting features have been affected by a higher jitter using VDE due to the synchronization between several UNIX processes. The scp, netcat and TFTP tests results show the times needed to transfer a 100MB file. TFTP suite has been tested on v4 only. Some performance lacks (e.g. TFTP) seems to be related to inefficient fragmentation on packets. The last test is on opening/closing/testing of network connections. The tool nmap is a port scan. In this case the loss in performance is quite high because several round trip messages are involved, and the latency is higher due to a larger number of IPC steps.

In Table II some preliminary tests on ale4net are presented. The test environment is composed by a Pentium 4 computer, 3.2Ghz, 1GB RAM, remotely connected through

program	no VDE	VDE v4 clear	VDE v4 encrypt	VDE v6 clear	VDE v6 encrypt
ping(v4)	0.127 ms	1.8 ms	2.7 ms	2.0 ms	1.7 ms
ping(v6)	0.466 ms	2.0 ms	3.7 ms	2.0 ms	0.7 ms
scp v4	9 s	24 s	36 s	11 s	53 s
scp v6	10 s	15 s	53 s	17 s	53 s
netcat v4	9.9s	12.2s	49.5s	11.3s	53s
netcat v6	10.6s	12.4s	40s	13.2s	47s
TFTP v4	34.7s	94.6s	216.9s	84.2s	189.6s
nmap v4	0.351s	4.0s	4.1s	4.8s	3.9s
nmap v6	0.361s	4.6s	4.5s	4.1s	10.4s

TABLE I  
SOME PERFORMANCE TESTS ON VDE

an ADSL line (640/120Kbps, typical latency 78ms) to a remote VDE tunnel broker (Dual Xeon 2.3 Ghz, 512MB). Two different shells (of the same user) have been connected to the same VDE tunnel broker: the first by VDE tunneling, the second by ale4net. The rows of Table II show the tests: 20 packet bursts of ping, data transfer (1MB file) by using scp and netcat and a portscan. It is clear from the results that the user level management of the protocol stack is quite time consuming especially for data transfer. Ale4net has not been optimized yet but the point is that regardless of the minor performances ale4net gives features that are not possible with standard tunneling: process level configuration of networking access with no need of root (administration) privileges.

program	VDE	Ale4net
ping(v4)	82.2 ms	86.5 ms
ping(v6)	84.7 ms	85.3 ms
scp v4	26 s	120 s
scp v6	26 s	123 s
netcat v4	18 s	124s
netcat v6	20 s	131s
nmap v4	144.6s	147.5s
nmap v6	155.4s	233.2s

TABLE II  
SOME PRELIMINARY TESTS ON ALE4NET

## XI. CONCLUSIONS AND FUTURE WORK

Virtual Square networks are multipurpose tools able to give a simple and generalized solution for a wide area of applications.

Virtual square solutions have been applied to create a virtual lab at the University of Bologna for the Practicum in Operating System (Laboratorio di Sistemi Operativi) Course, spring term 2004. Students were able to configure, administer and run their own operating systems, to access the Internet from them and to provide services on the virtual network, to join the network and work from home. The solution is safe: students do not have any root access or configuration rights on real networks and systems of the Department. There is a VDE tunnel broker at the university that logs all the virtual cables and the mapping between IP addresses in the real Internet and IP traffic running on the

cable. In this way it is possible to track eventual abuses. The code for this functionality is part of the vde plug application.

I personally use VDE almost daily to join IPv6 networks with the ADSL and GPRS services while the Italian cellular phone and Internet providers do not give any IPv6 service yet.

This year several other services are going to start. We have designed a Virtual Lab where students can test a lot of different Operating Systems (see the FreeOSZoo project [21]). This virtual lab will be used by students in Computer Science and by students of the new Master in Free and Open Source Software Technology.

Several new features are under study.

- **management of loops in VDE networks.** The minimum spanning tree used for switches in a LAN is not effective for a geographically distributed extended LAN with bandwidth bottlenecks. Some kind of level 2 routing is under investigation.
- **management of IPv4 and IPv6 multicast streams.** Multicasts are currently managed as broadcasts, it works but very inefficiently.
- **Ale4net** is not complete. A new feature for the user choice of the DNS server to be used for name resolving has been completed and will be released soon. Support for PF\_PACKET sockets is under development to move services like DHCP client/server at user level. IPv6 does not support address autoconfiguration yet.

## XII. ACKNOWLEDGEMENTS

The implementation of vde\_switch includes some code from the User-Mode Linux networking tool by Yon Uriarte and Jeff Dike. Ale4Net includes code from Alpine4Linux by Neelkanth Natu and from lwip by Adam Dunkels. This work is based on Free and Open Source Software. It would have been very hard, if not impossible, to do something similar based on proprietary solutions: requiring a host of non disclosure agreements, guarantees regarding access to interesting source code, possibly a promise to avoid integrating code owned by competitors, and the asking of permission to publish the research results.

I feel that today Free and Open Source software is the key for authentic, open minded, long term research in computer science. Therefore I want to use the opportunity to thank the whole community that develops, debugs and broadcasts Free and Open Source Software and ideas.

I would like to acknowledge my colleague Michael Goldweber of the Xavier University for his support and his precious cooperation.

## XIII. REFERENCES

- [1] F. Ballard. Qemu project home page. <http://fabrice.bellard.free.fr/qemu/>.
- [2] R. Davoli. Virtual square home page. <http://www.virtualsquare.org/>.
- [3] Renzo Davoli. Ale4net sourceforge home page. <http://ale4net.sourceforge.net>.
- [4] Renzo Davoli. Vde sourceforge home page. <http://vde.sourceforge.net>.
- [5] J. D. Dike. User-mode linux. In *Proc. of 2001 Ottawa Linux Symposium (OLS)*, Ottawa, 2001.
- [6] J. D. Dike. Making linux safe for virtual machines. In *Proc. of 2002 Ottawa Linux Symposium (OLS)*, Ottawa, 2002.
- [7] Adam Dunkels. Lwip web page. <http://www.sics.se/~adam/lwip/>.
- [8] Adam Dunkels. Minimal tcp/ip implementation with proxy support. Master's thesis, SICS - Swedish Institute of Computer Science, February 2001.
- [9] David Ely, Stefan Savage, and David Wetherall. Alpine: A user-level infrastructure for network protocol development. In *Proc. of 3rd USENIX Symposium on Internet Technologies and Systems (USITS01)*, March 2001. San Francisco.
- [10] Sebastian Biallas et al. Pearpc home page. <http://pearpc.sourceforge.net>.
- [11] J. Gilmore, H. Daniel, M. Richardson, R. G. Briggs, H. Redelmeier, C. Schmeing, S. Sgro, J. Ioannidis, A. D. Keromytis, H. Spencer, and S. Harris. Free s/wan project web site.
- [12] K. Hamzeh, G.S. Verstein, W. Vertheini, J. tarraudi, and W.A. Point-to-point tunneling protocol. *Internet draft, IETF*, July 1997.
- [13] R. Hinder and S. Deering. *RFC 1884 - IP Version 6 Addressing Architecture*. IETF, 1995.
- [14] S. Kent and B. Atkinson. Rfc 2041: Security architecture for the internet protocol. Technical report, IETF, 1998.
- [15] M. Krasnyansky. Virtual tunnel: Vtun (web site). <http://vtun.sourceforge.net>.
- [16] B. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10), October 1973.
- [17] K. Lawton. Bochs project home page. <http://bochs.sourceforge.net>.
- [18] M. A. Lehmann. Virtual private ethernet (vpe) web site. <http://savannah.gnu.org/projects/gvpe>.
- [19] M. Morsiani and R. Davoli. Learning operating system structure and implementation through the MPS computer system simulator. In *Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education*, pages 63–67, New Orleans, 1999.
- [20] Neelkanth Natu. Alpine4linux. <http://www.vzavenue.net/~neelnatu/alpine4linux/>.
- [21] Jean-Michel Pouré and Renzo Davoli et al. Free operating systems zoo. <http://www.freeoszoo.org>.
- [22] HoneyNet Project. Know your enemy: Defining virtual honeynets. <http://www.honeynet.org/papers/virtual/index.html>.
- [23] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. *RFC 2661 - Layer Two Tunneling Protocol "L2TP"*. IETF, 1999.
- [24] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the 5th Symposium on Operating Systems Design & Implementation*, pages pp. 255–270, December 2002. Boston.
- [25] J. Yonan. Openvpn project web site. <http://openvpn.sourceforge.net>.
- [26] Zebedee web site. <http://www.winton.org.uk/zebedee/>.