



# More than Virtual: Virtual Square!

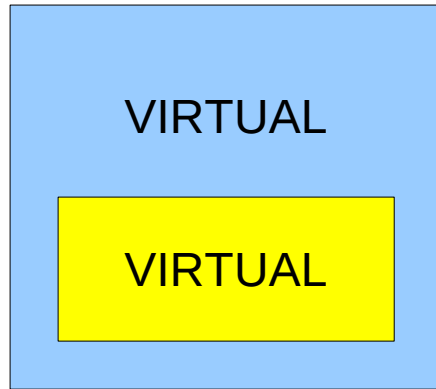
Renzo Davoli  
Computer Science Department  
ALMA MATER STUDIORUM: University of Bologna

Linux Day

October, 25 2008

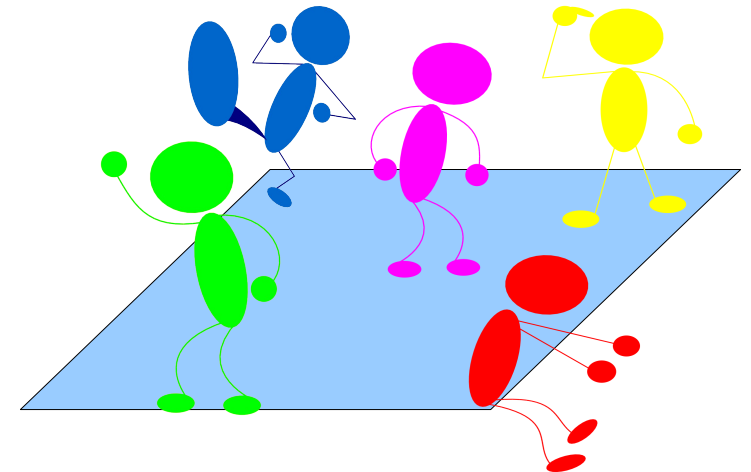
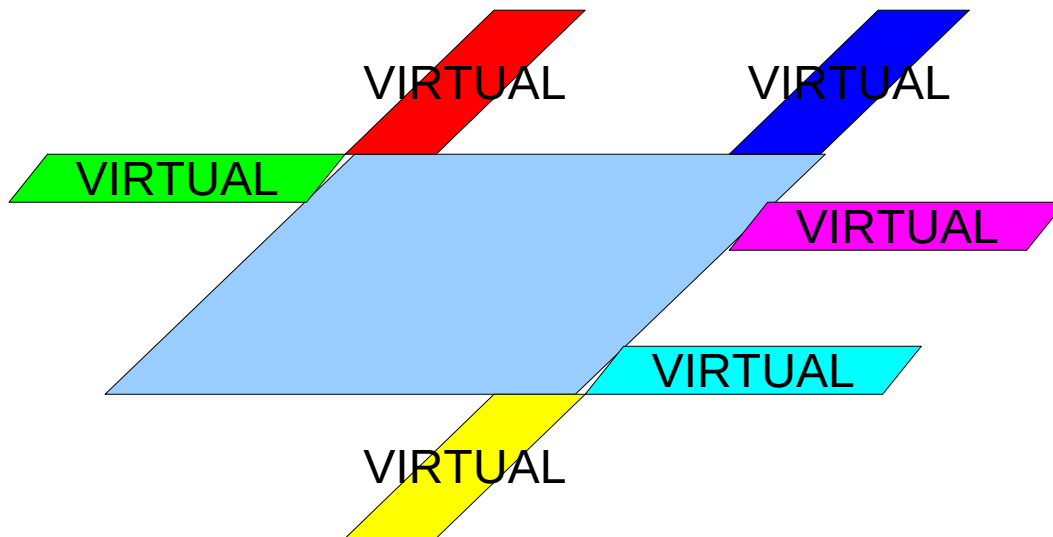


# Virtual Square



VIRTUAL SQUARED

VIRTUAL SQUARE





# VIRTUALITY today

- Virtual Machines
  - historical topic
  - lots of papers
  - lots of tools
  - ... but something is already missing
- Virtual Networking
  - less historical
  - several papers



# Virtual Square

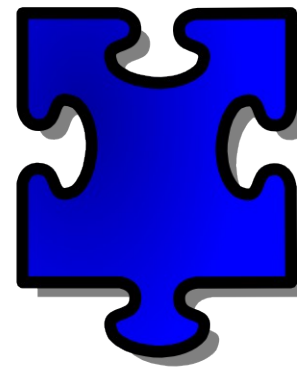
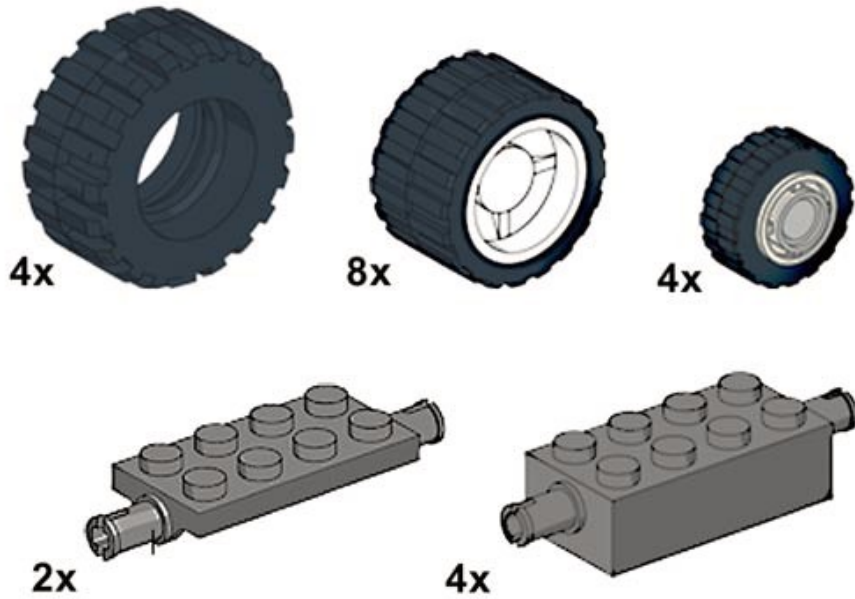
Virtualization concepts and tools are disconnected.

There is a world of new applications that can be realized by interoperating, integrated virtuality

**UNIFICATION IS NEEDED**

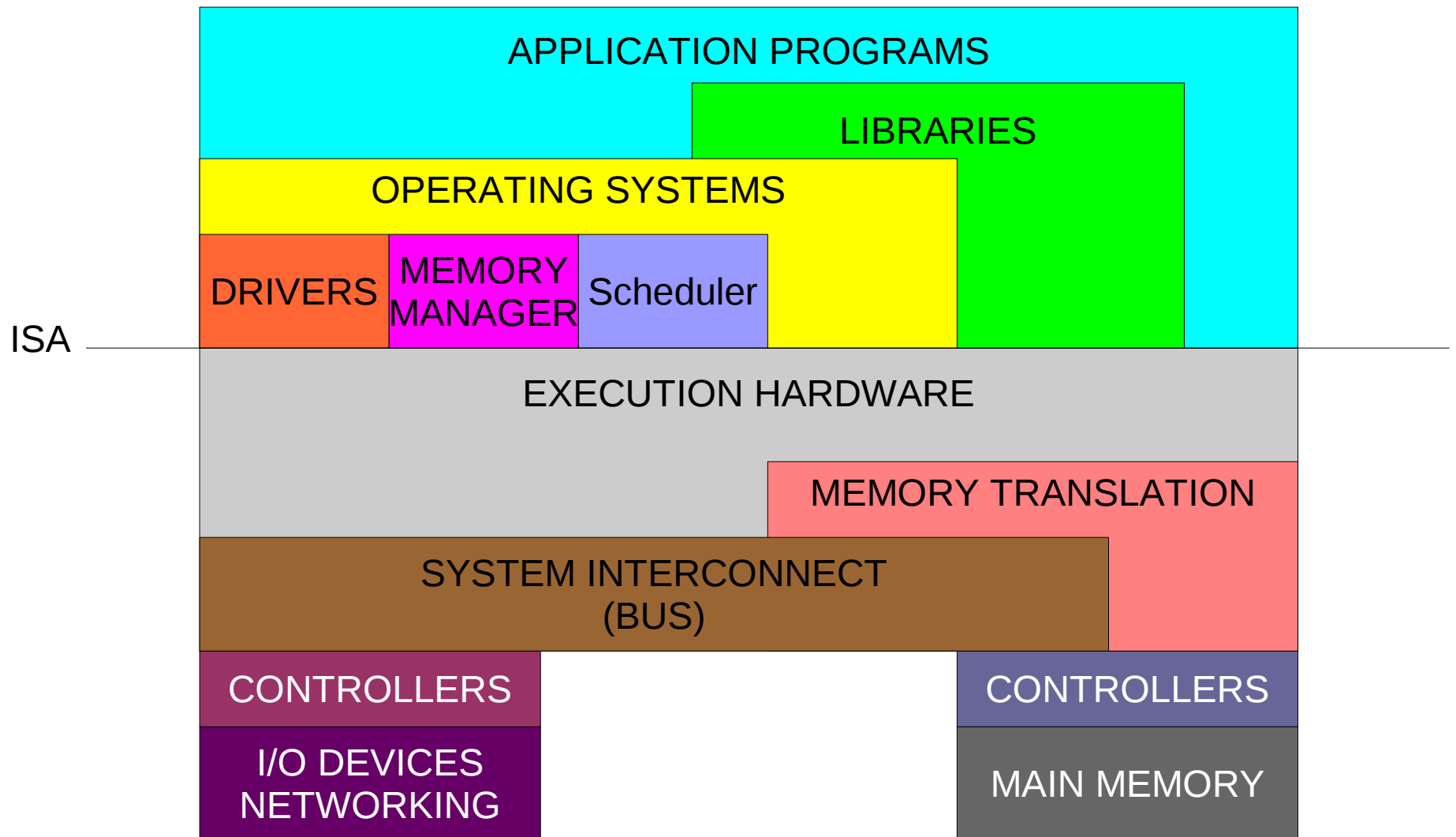


# Virtual Square



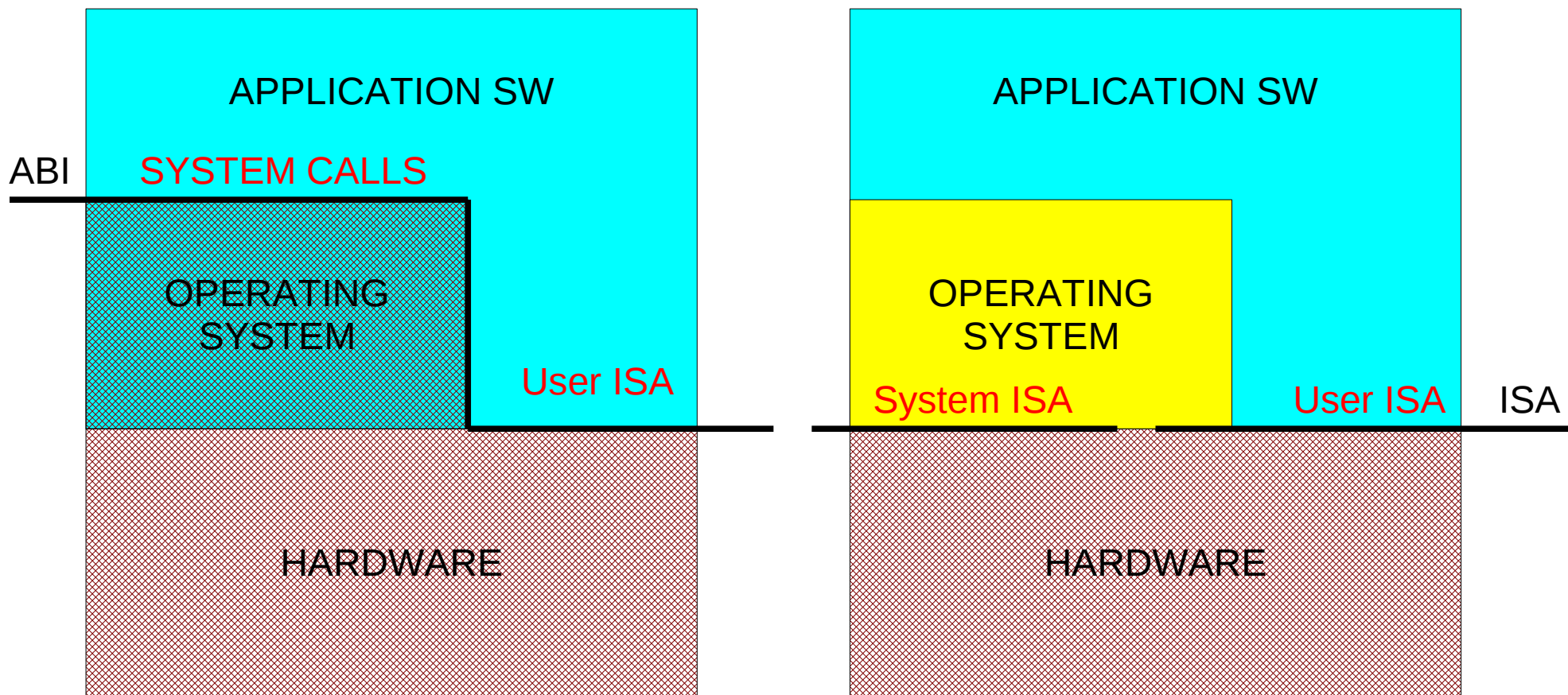


# Classical Theory of VM Computer Architecture (Myers)





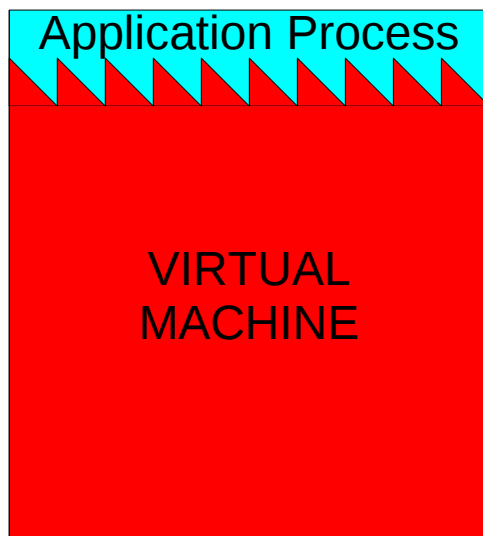
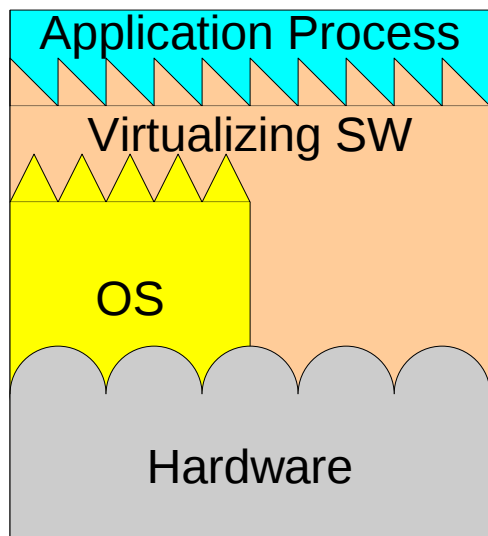
# ABI vs ISA (Smith Nair 2005)





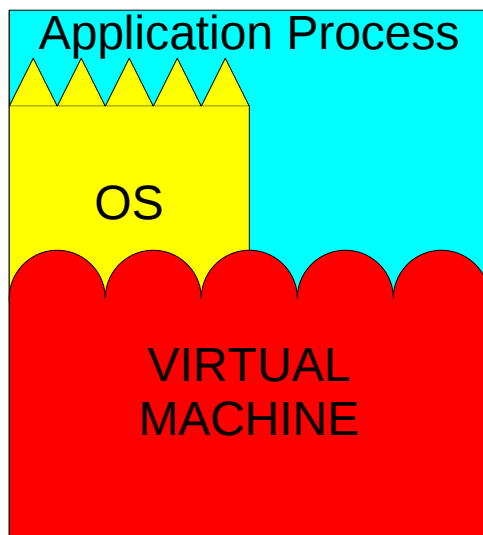
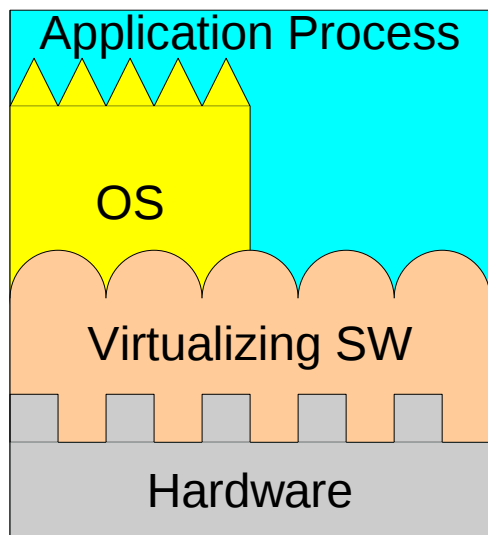
# Classical Theory of VM

## Process / Machine VM (Smith-Nail 2005)



PROCESS VM (PVM):

- one process
- a.k.a. runtime
- PVM provides ABI

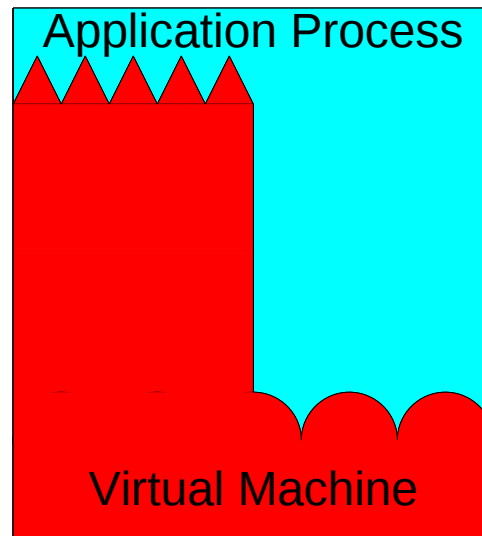
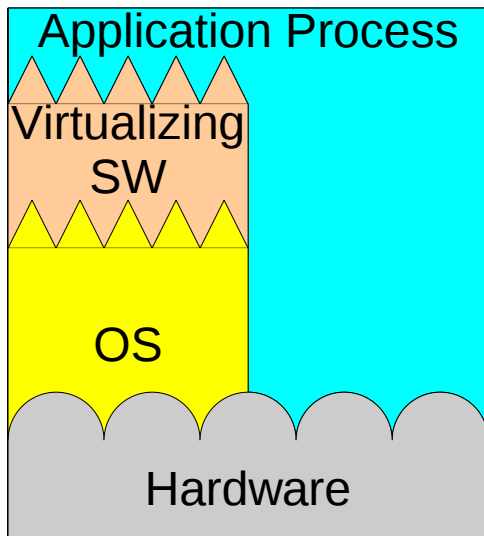


SYSTEM VM (SVM):

- SVN provides ISA
- SVN needs an OS



# NEW Category: Syscall Virtual Machine (SCVM)



System Call VM:

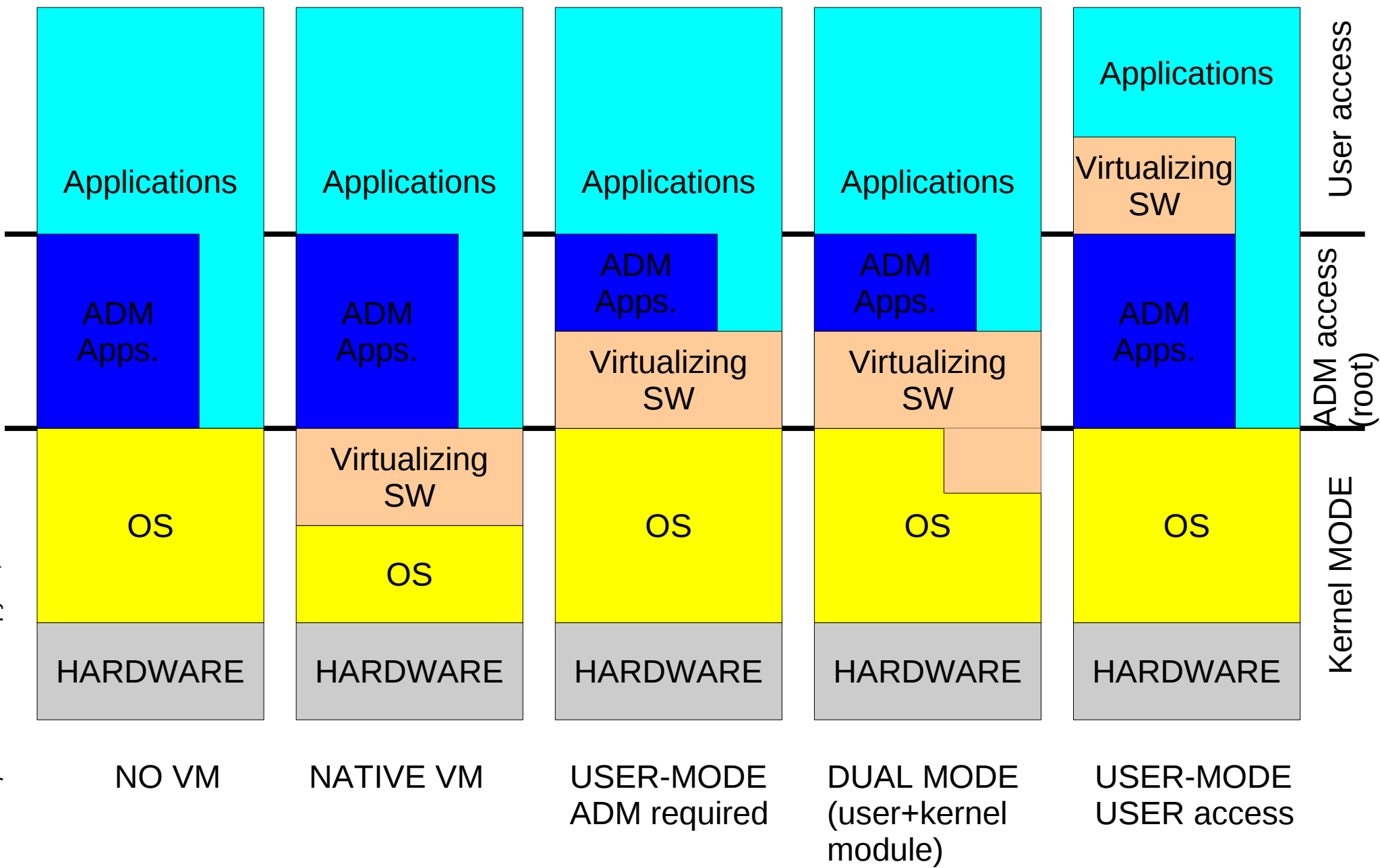
- Same ISA
- Maybe same ABI
- Syscall trapped

Novil 2005, pag. 410: "Virtualizing at the System Call Level:  
the virtualization process could be made more efficient by intercepting the initial I/O  
at the OS interface, the ABI... In general this is a rather daunting task (creating a SCVM)  
done only if the guest OS is well structured and understood intimately by the SCVM

THEORY", in practice!



# Mode/Permission





# Some Examples of VM (free software)

- Qemu: PVM or SVM, User Mode User Access (or dual-mode with KQEMU, proprietary sw).
  - cross emulation platform (ia32, ia64, ppc, m68k, sparc, arm...)
  - dynamic translation
- XEN: SVM, Native.
  - xen uses para-virtualization (O.S. in domain0 has the real device drivers).
  - (xen ideas come from the Denali project: SVN, Native, real virtualization).



# Some Examples of VM (free software)

- User-Mode Linux (U-ML): SCVM, User-Mode User Access.
  - Same ABI/Same ISA: Linux on Linux.
  - Uses the debugging interface ptrace to capture the system calls.
- OpenVZ: Native, SVM (but all the Vms share the same kernel, this category is also known as: Operating System Level Virtualization).
  - derived from the proprietary product Virtuozzo.
  - the kernel itself provide different resource and different permissions to virtual instances of Linux.



# Some Other Examples

- Mac-On-Linux: (free SW, SVM, Dual)
- Vmware: (proprietary, SCVM, Dual)
- VirtualBOX: (dual licencing, SCVM, Dual)
- VirtualPC/Virtual Server: (proprietary, SVM, Dual)
- PearPC: (free, SVM, User-Mode User-Access)
- Solaris “zones” (aka containers): proprietary, Native, SVN (OSLV).



# There are other kinds of Virtuality!

The “classical” definition of Virtual Machines does not catch all the possible “Virtualities” ...

- Virtual Network Interfaces:
  - tuntap
- Virtual Networks:
  - VPN, Local VN provided with Virtual Machines
- Virtual Running Environments:
  - chroot (POSIX syscall), fakeroot utility UNIX, Linux Vserver
- Virtual File Systems
  - Fuse, UnionFS, PlasticFS (uses LD\_PRELOAD)



# Other “Virtuality”

- Environmental Subsystems.
  - system call conversion: e.g. Posix -> Win32
- System Call Interposition.
  - Systrace: system call access manager
  - Janus, Ostia: application firewall.



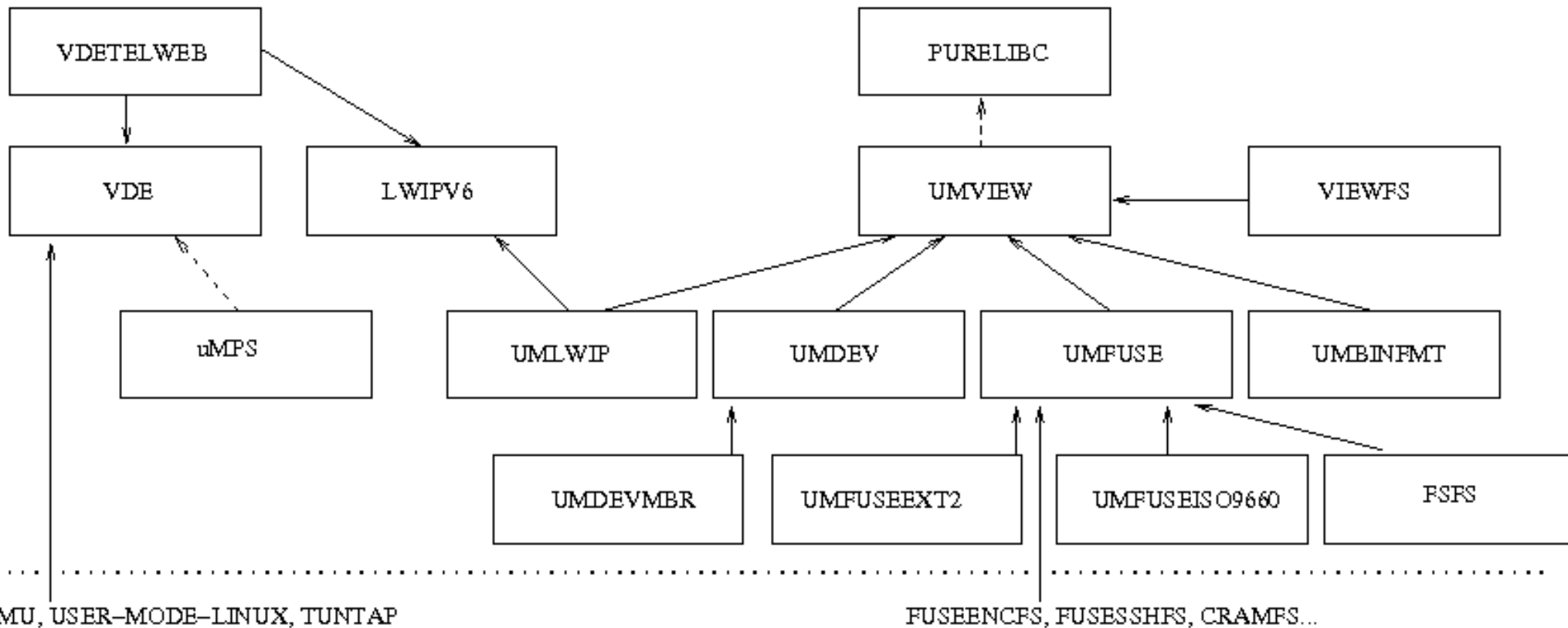
# Virtual Square Goals

- We have seen a wide (wild ;-)) world of different kinds of virtuality based on different ideas, several tools
- Keyword #1: communication
  - different VMs must be interconnected
- Keyword #2: integration
  - different VMs can be seen as special cases of a broaden idea of VM
- Keyword #3: extension
  - several needs could be captured by some VM abstraction, but there is not the specific model of VM, nor the tool!



# 2 1/2 Years!

- Ideas, tools... more than 100,000 lines of free SW.



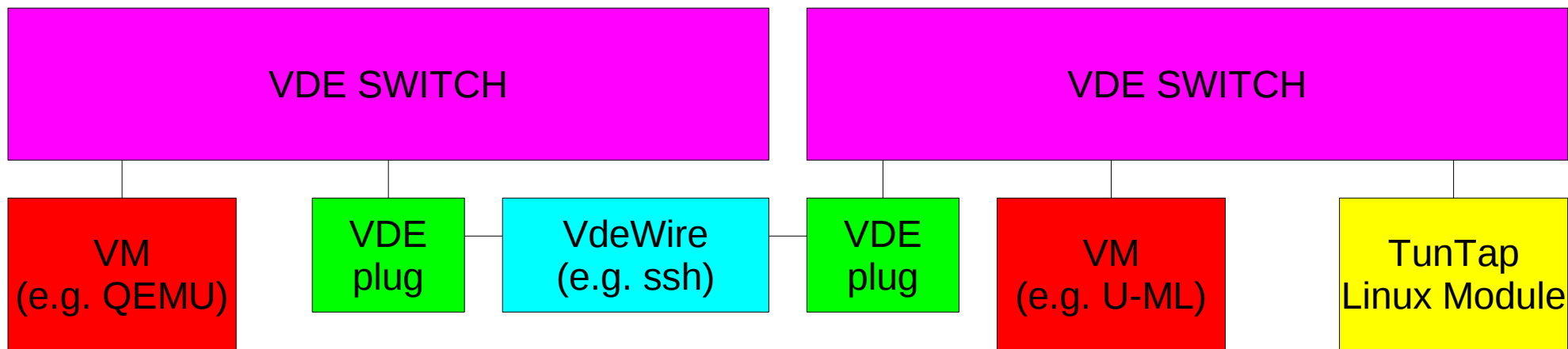
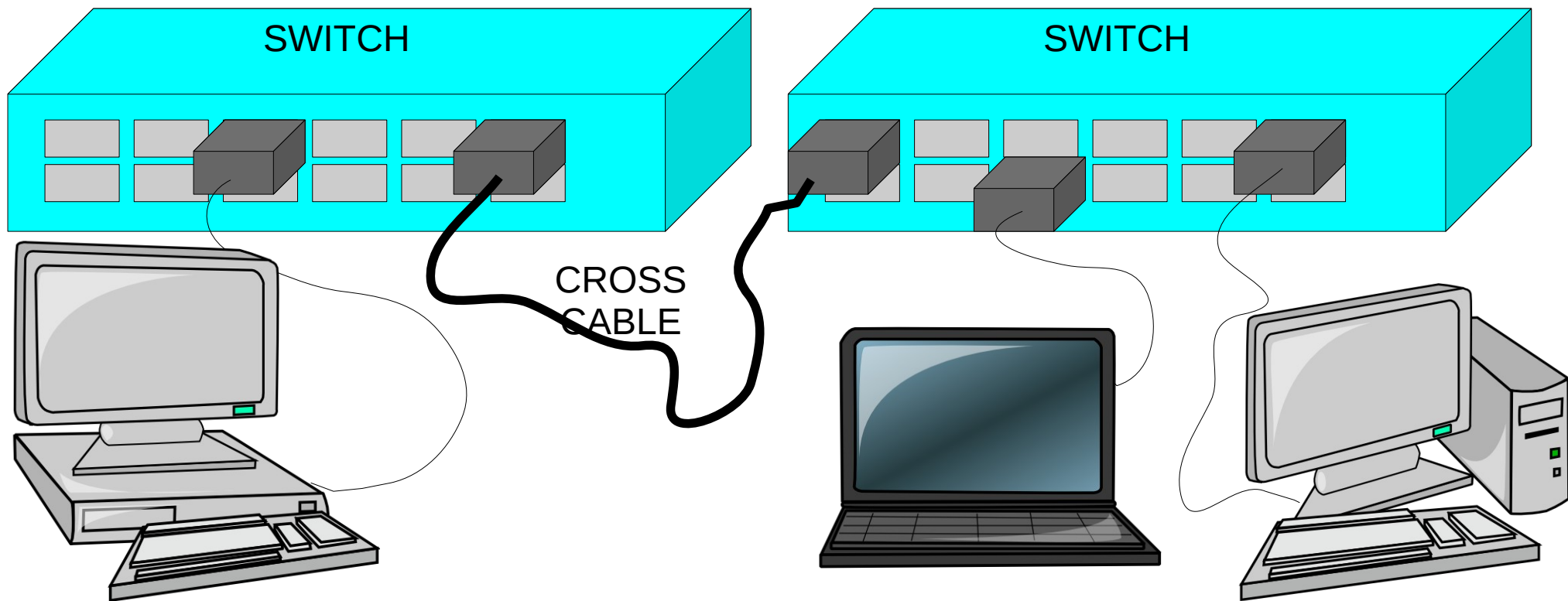


# Keyword #1: Communication

- Virtual Distributed Ethernet:
  - VDE clients: User-Mode Linux, Qemu, virtual tap (all appl. using tap), tuntap (in linux kernel), slirp.
  - VDE appears as a Local Area Network (Ethernet complaint) from all the connected entities.
  - The machines interconnected by a VDE may be distributed on different hosts (no limits of “distance”, km or hops).



# VDE components





# VDE: Related Work

- VPN: (OpenVPN) point2point, for real machines
- Overlay Networks: specific for application (peer to peer, Akamai).
- VM networking: (tools provided with VM, e.g. uml-switch) specific for VM

## VDE:

- multipoint, general mesh
- no need for root (administration) access
- heterogeneous VM and non VM connected



# VDEv2: advertisement

- VDEv2:
  - modular design
  - compatible with user-mode linux, qemu, tuntap, (bochs, plex86), umview/lwipv6
  - through the vdetaplib potentially compatible any application using tap
  - VLAN (802.1Q)
  - FST (fast spanning tree)
  - run time maneageable via unixterm (telnet or web with vdetelweb)
  - includes slirpvde and wirefilter
  - status debug (NEW!)
  - plugin support (NEW!)



# VDEv2

- VDE-Switch
  - number of ports configurable on command line
  - port0 is reserved for management clients, n-1 ports are available for connections.
  - management UNIX socket for management clients
    - self-describing SMTP-like protocol
  - modules: datasock (VM conn), tuntap, consmngmt (management)



# VDE cables

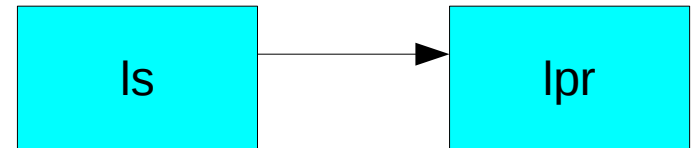
- VDE-plug
  - is a VM that converts the Ethernet packets of a VDE port into a stream connection (stdin-stdout)
- VDE-wire
  - can be any application able to give a stdin/stdout stream connection



# Dual Pipe

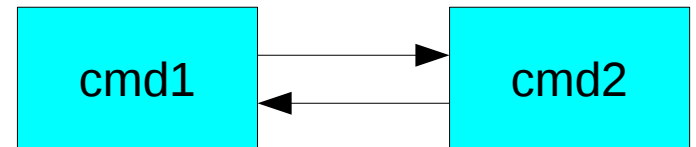
- dpipe is a new (general purpose) command we have added.
- Pipe are well known abstractions. The following command prints the list of the current directory:

ls | lpr



- Dpipe creates a bi-directional connection between the processes

dpipe cmd1 = cmd2





# VDE cables, plugs, wires and dpipe

- dpipe is used to create VDE-cables:

```
dpipe vde_plug = ssh vde.students.cs.unibo.it vde_plug
```

- this command connects by a dpipe the local vde\_plug with a vde\_plug running on a remote host (the wire is ssh)
- other applications can be used as wire (e.g.netcat)
- In the example vde\_plug refers to the default switch. It is possible to run several switches on the same host, an extra option is needed in this case.



# wirefilter

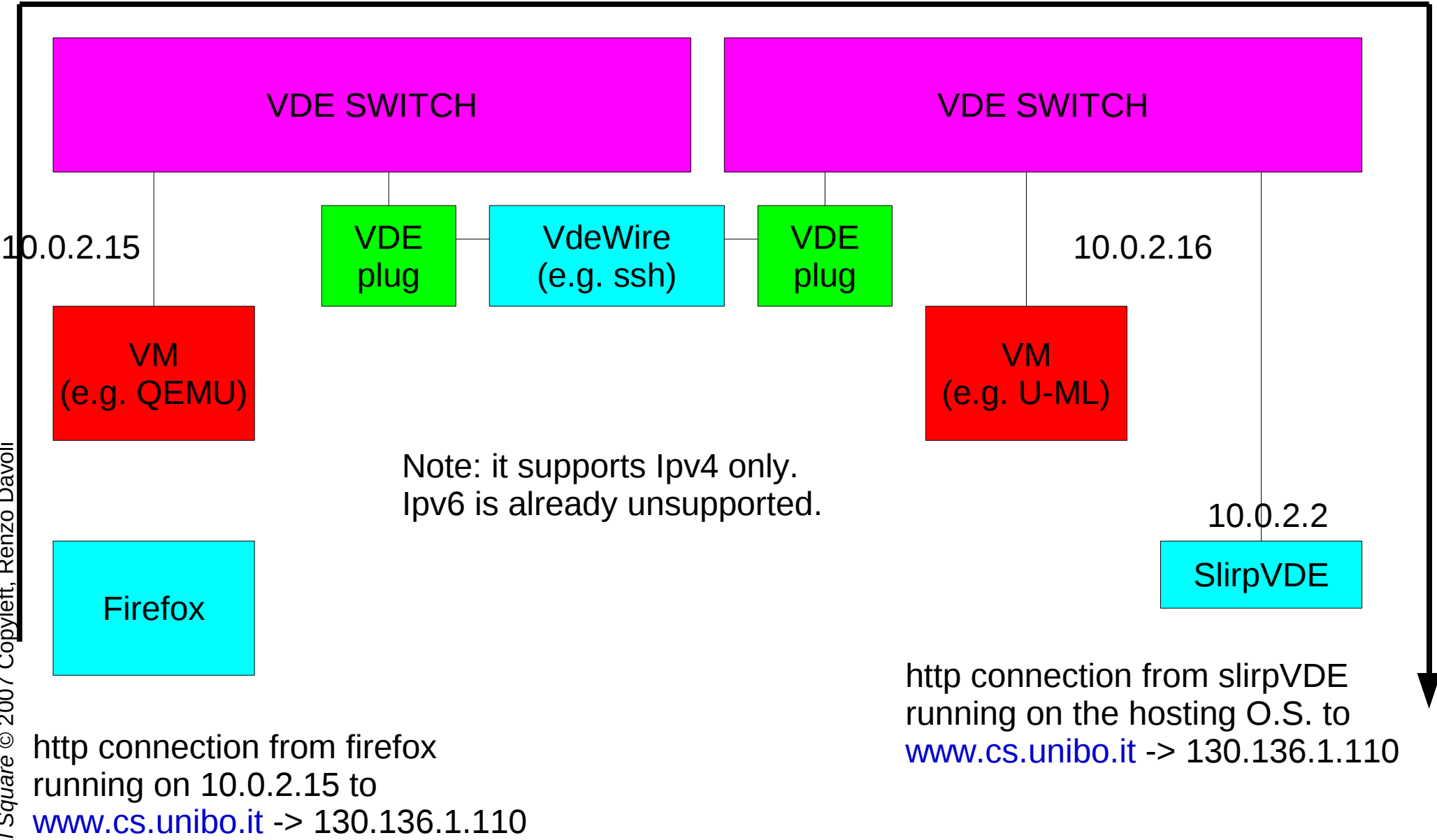
- wirefilter can be put on a cable (e.g. for network testing)

`dpipe vde_plug /tmp/s1 = wirefilter -m /tmp/m = vde_plug /tmp/s2`

- packet loss, delays, dup, speed, noise figures, mtu, fifo properties of the line can be changed with command line options or real time via a management socket.



# SlirpVDE



Note: it supports Ipv4 only.  
Ipv6 is already unsupported.

http connection from firefox  
running on 10.0.2.15 to  
[www.cs.unibo.it](http://www.cs.unibo.it) -> 130.136.1.110

http connection from slirpVDE  
running on the hosting O.S. to  
[www.cs.unibo.it](http://www.cs.unibo.it) -> 130.136.1.110



# vde\_cryptcab

- Coded by Daniele Lacamera (danielinux)
- A vde\_cryptcab is a distributed cable manager for VDE switches.
- Server side

```
vde_cryptcab -s /tmp/vde2.ctl -p 2100
```
- Client side

```
vde_cryptcab -s /tmp/vde2.ctl -c foo@remote.machine.org:2100
```
- use a blowfish channel (random key exchanged by scp).



# LWIPv6

- It is a LWIPv4/v6 stack implemented as a library.
- Fork project from LWIP project (Adam Dunkels <[adam@sics.se](mailto:adam@sics.se)>)
- Can be connected to any number of VDE, TUN, TAP interfaces.
- It is a hybrid stack (not a dual-stack). One single Ipv6 “engine” is able also to manage Ipv4 packets in compatibility mode (130.136.1.110 is managed as 0::ffff:130.136.1.110).



# LWIPv6

- PF\_INET, PF\_INET6
- PF\_PACKET for raw packet management
  - support for user-level network analysis tools (e.g. sniffers, ethereal)
  - support for user-level dhcp clients.
- PF\_NETLINK for configuration
- Packet filtering



# LWIPv6 interface definition API

```
struct netif *lwip_vdeif_add(void *arg);  
struct netif *lwip_tapif_add(void *arg);  
struct netif *lwip_tunif_add(void *arg);
```

```
int lwip_add_addr(struct netif *netif, struct ip_addr *ipaddr, struct ip_addr  
*netmask);
```

```
int lwip_del_addr(struct netif *netif, struct ip_addr *ipaddr, struct ip_addr *netmask);
```

```
int lwip_add_route(struct ip_addr *addr, struct ip_addr *netmask,  
                  struct ip_addr *nexthop, struct netif *netif, int flags);
```

```
int lwip_del_route(struct ip_addr *addr, struct ip_addr *netmask,  
                  struct ip_addr *nexthop, struct netif *netif, int flags);
```

```
int lwip_ifup(struct netif *netif);  
int lwip_ifdown(struct netif *netif);
```



# LWIPv6 socket API (just put lwip\_ ahead)

```
int lwip_accept(int s, struct sockaddr *addr, socklen_t *addrlen);
int lwip_bind(int s, struct sockaddr *name, socklen_t namelen);
int lwip_shutdown(int s, int how);
int lwip_getpeername (int s, struct sockaddr *name, socklen_t *namelen);
int lwip_getsockname (int s, struct sockaddr *name, socklen_t *namelen);
int lwip_getsockopt (int s, int level, int optname, void *optval, socklen_t *optlen);
int lwip_setsockopt (int s, int level, int optname, const void *optval, socklen_t optlen);
int lwip_close(int s);
int lwip_connect(int s, struct sockaddr *name, socklen_t namelen);
int lwip_listen(int s, int backlog);
int lwip_recv(int s, void *mem, int len, unsigned int flags);
int lwip_read(int s, void *mem, int len);
int lwip_recvfrom(int s, void *mem, int len, unsigned int flags,
    struct sockaddr *from, socklen_t *fromlen);
int lwip_send(int s, void *dataptr, int size, unsigned int flags);
int lwip_sendto(int s, void *dataptr, int size, unsigned int flags,
    struct sockaddr *to, socklen_t tolen);
int lwip_socket(int domain, int type, int protocol);
int lwip_write(int s, void *dataptr, int size);
int lwip_select(int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset,
    struct timeval *timeout);
int lwip_ioctl(int s, long cmd, void *argp);
```



# VDETELWEB

- It is the Web/Telnet Server for VDE switch configuration.
- It uses the LWIPv6 library
- It has two connections to the controlled VDE switch:
  - management socket to give commands
  - port0: the ethernet port used by the TCP-IP stack.
- It reads the set of commands, descriptions, arguments from the switch itself.
- Telnet has history/command editing and support for asynch debug output (NEW)



# VDETELWEB: telnet

```
xterm
/
renzo@titanic2:/$ telnet 192.168.250.5
Trying 192.168.250.5...
Connected to 192.168.250.5.
Escape character is '^]'.
VDE switch V.2.0.3
(C) R.Davoli 2005 - GPLv2

Login:[SSL not available]
                               admin

Password:
VDE2@titanic2[/var/run/vdectl]: port/print
Port 0001 untagged_vlan=0000 ACTIVE - Unnamed Allocatable
  -- endpoint ID 0007 module tuntap      : tap0
Port 0002 untagged_vlan=0000 ACTIVE - Unnamed Allocatable
  -- endpoint ID 0009 module unix prog   : LWIPv6 user=root PID=31639 SOCK=/var
/run/vdectl.31639-00
Success

VDE2@titanic2[/var/run/vdectl]:
VDE2@titanic2[/var/run/vdectl]:
VDE2@titanic2[/var/run/vdectl]: █
```



# VDETELWEB: Web Interface

VDE2@titanic2[/var/run/vde.ct1]: hash - Firefox

File Edit View Go Bookmarks Tools Help

http://192.168.250.5/hash.html

The Mozilla Organi... Latest Builds VOGLIO BREVETTA... Corso di Sistemi O... Internet Banking

## VDE2@titanic2[/var/run/vde.ct1]: HASH TABLE MENU

|                           | Syntax     | Args                 | Description                       |
|---------------------------|------------|----------------------|-----------------------------------|
| <a href="#">Home Page</a> |            |                      |                                   |
| <a href="#">ds</a>        |            |                      |                                   |
| <a href="#">hash</a>      |            |                      |                                   |
| <a href="#">fstp</a>      |            |                      |                                   |
| <a href="#">port</a>      |            |                      |                                   |
| <a href="#">vlan</a>      |            |                      |                                   |
| showinfo                  |            |                      | show hash info                    |
| setsize                   | N          | <input type="text"/> | change hash size                  |
| setgcint                  | N          | <input type="text"/> | change garbage collector interval |
| setexpire                 | N          | <input type="text"/> | change hash entries expire time   |
| setminper                 | N          | <input type="text"/> | minimum persistence time          |
| print                     |            |                      | print the hash table              |
| find                      | MAC [VLAN] | <input type="text"/> | MAC lookup                        |

**VDE2@titanic2[/var/run/vde.ct1]: hash/print**

```
Hash: 0036 Addr: 02:02:32:f2:40:06 VLAN 0000 to port: 002 age 0 secs
Hash: 0124 Addr: ae:b8:8a:6c:48:92 VLAN 0000 to port: 001 age 0 secs
```

**Result: Success**

---

VDE 2.0 WEB MGMT INTERFACE

Done



# Keywords #2/#3: Integration and Extension

## • Questions:

- is there a model able to capture several different types of “virtuality”?
  - File System virtuality (Virtual FS: FUSE, chroot, unionfs, chroot, fakeroot, etc)
  - Networking virtuality (VPN, VDE)
  - ...
- are there practical problems that cannot be solved with current models of Virtual Machines?



# View-OS

–A process with a view...



# View-06



mount

sshd

firefox

so

ssh

office

postfix

httpd

SILVIA DAVOLI



# View

- Each process has a view of the “world”.
  - meaning of a path
  - IP address, routing
  - ...
- Global View assumption: all the processes running on the same computer share the same view.
- (Almost) true in POSIX. Some notable exceptions: chroot and... virtual machines.



# View and Virtual Machines

- SVM require the boot of an entire OS.
- User-Mode Linux is a SCVM, but it boots a Linux kernel.
- Fakeroot, chroot, FUSE, systrace etc. are specific virtualizations.
- chroot, FUSE, tuntap etc. require root access to run



# Global effects of commands

- A command has a global effect when it changes the view of all the processes running on the same machine.
  - e.g. mount of a filesystem
  - e.g. changing the IP address
- Some operations are forbidden to ordinary users just because the operating system is not able to give local effects to commands:
  - e.g. mount of a disk image
  - e.g. define a VPN for a user
  - e.g. change IP address/routing for a process.



# View-OS breaks the Global View Assumption

- Each process can have its own “view” of the world.
  - mount of filesystems
  - redefinition of access permissions to resources
  - definition of interfaces/IP addresses/routing
  - definition of devices
  - ...



# View-OS Goals

- Redefinition of System Call behavior
- Definition of new System Calls
- Binary Compatibility with existing software
- Modularity and Composition of local view definition
- Non-privileged use.



# View-OS and security

- A process can change its view with no security risks.
  - local mount of a disk image. It is just a sophisticated access to a file!
  - definition of a local VPN. The virtual interface used by the process is just a network connection for the hosting computer
- All these operation can be done by running a SVM or User-Mode Linux.
  - no extra risks,
  - waste of resources and time: boot of an entire kernel!



# View-OS and Security (the other way round)

- Loopback mount of a personal disk image (with private data on it) need to set carefully the access permissions (as we'd want a local effect but the kernel can only provide a global mount)
- Testing an unknown program/browsing dangerous edges of the internet can be very risky. Most hackers have “fake” user accounts to do such experiments. The definition of a very limited view of the filesystem/networking around the program/browser could act like a sandbox.

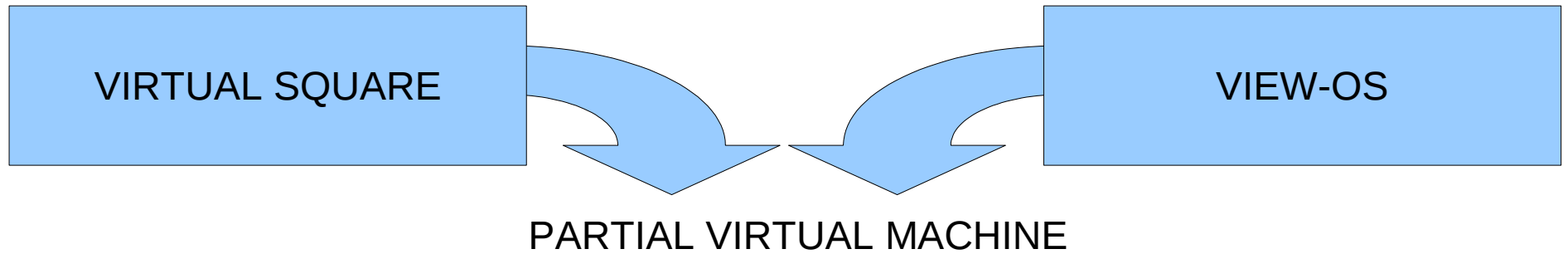


# View-OS implementation

- Specific Kernel:
  - need a lot of time to reach a usable level
  - need even more time to reach a suitable level of reliability.
- Virtual Machine
  - this is a new model of virtual machine: Partial Virtual Machines (ParVM)
- Kernel Module (with suitable support from the kernel)
  - intermediate approach (utrace made it possible)



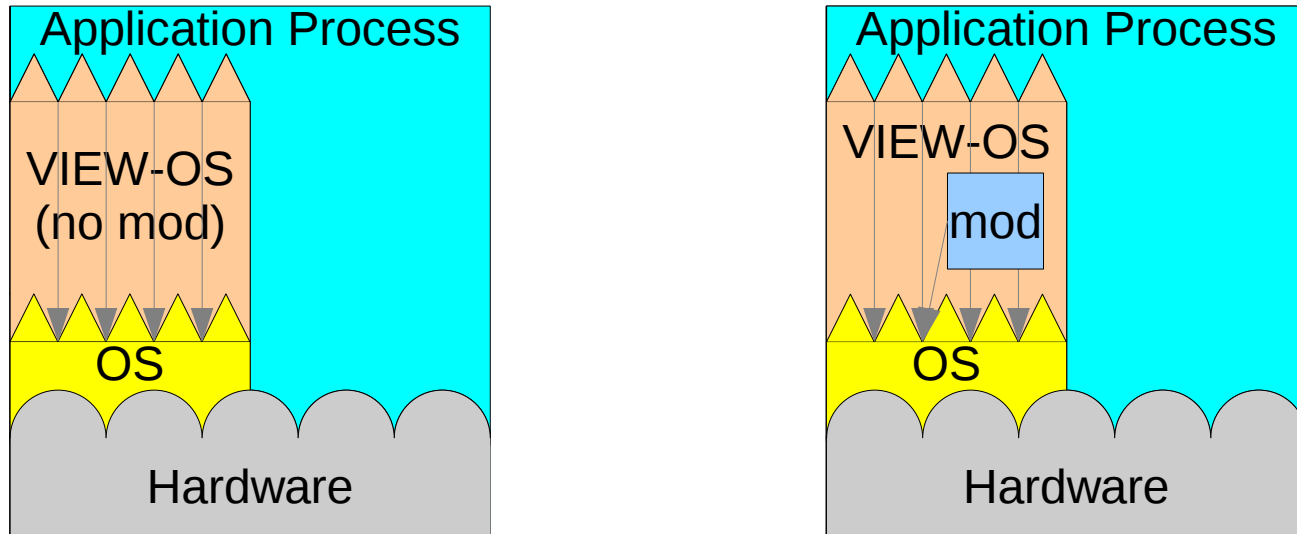
# Partial Virtual Machine (ParVM)



- A Partial Virtual Machine
  - support the same ISA of the hosting machine
  - support the same ABI of the hosting machine
  - can redefine some calls.
- ParVM can be composed.



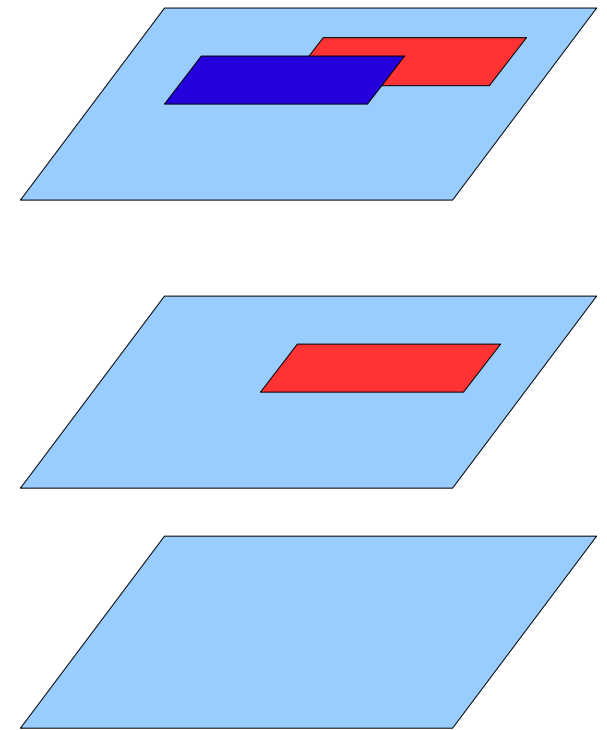
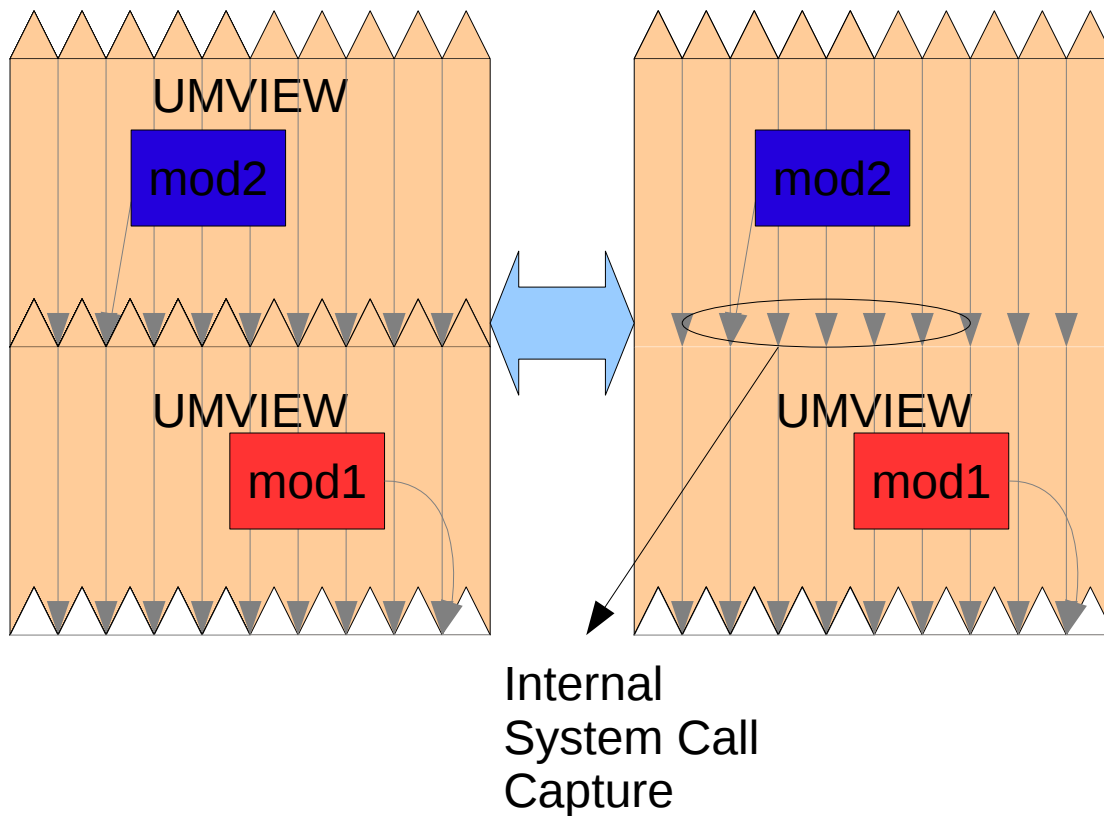
# UMVIEW=VIEW-OS as a ParVM



- UMVIEW is a generic ParVM (SCVM).
  - UMVIEW load modules for specific virtualizations
  - UMVIEW running without modules is an empty ParVM (each call is directly mapped on the same call of the hosting machine).



# Composition of Modules



- Modules can be composed.
- It is like superimposing overlays
- Module composition can be done by one instance of umview. Internal call must be captured (PURELIBC)



# UNIX is file system centric

- Pathname = Global Naming Scheme
- /dev, fifo, UNIX socket, /proc
- Advantages:
  - no specific system calls
  - inheritance of many “methods”, e.g. access prot.
- Overlay of file system subtrees = redefinition of what got the name from the subtree
- Basic operation: mount



# UMVIEW mount

- Mount is a “view transformation” function:

$$v_1 = m_1(v_0)$$

- A further mount “sees” the modified view

$$v_2 = m_2(v_1) = m_2(m_1(v_0)) = m_2 \circ m_1(v_0)$$

- UMVIEW can mount an image hidden by the mount operation itself
- UMVIEW can mount files (not only directory)
- UMVIEW mount can change the view outside the mountpoint
- UMVIEW can mount on a non-existing mountpoint

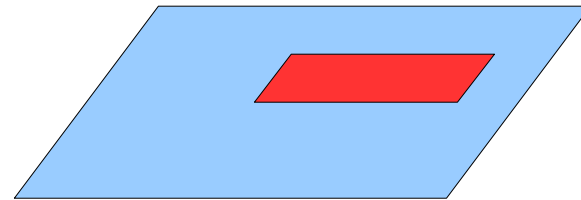
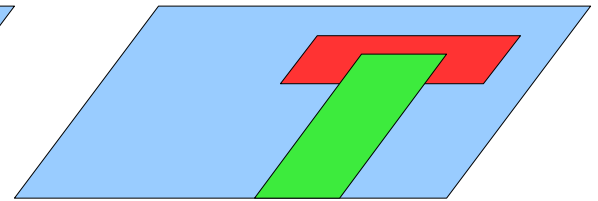
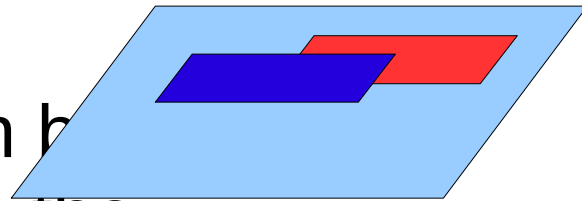


# Nested UMVIEW

- What happens when a user starts UMVIEW into UMVIEW?

umview xterm &

- At the time of activation by ParVM instances share the view, then each one can do further “mount”s
- It is a “virtual” instance: the same UMVIEW take into account the “view split” (by the treepoch data struct).



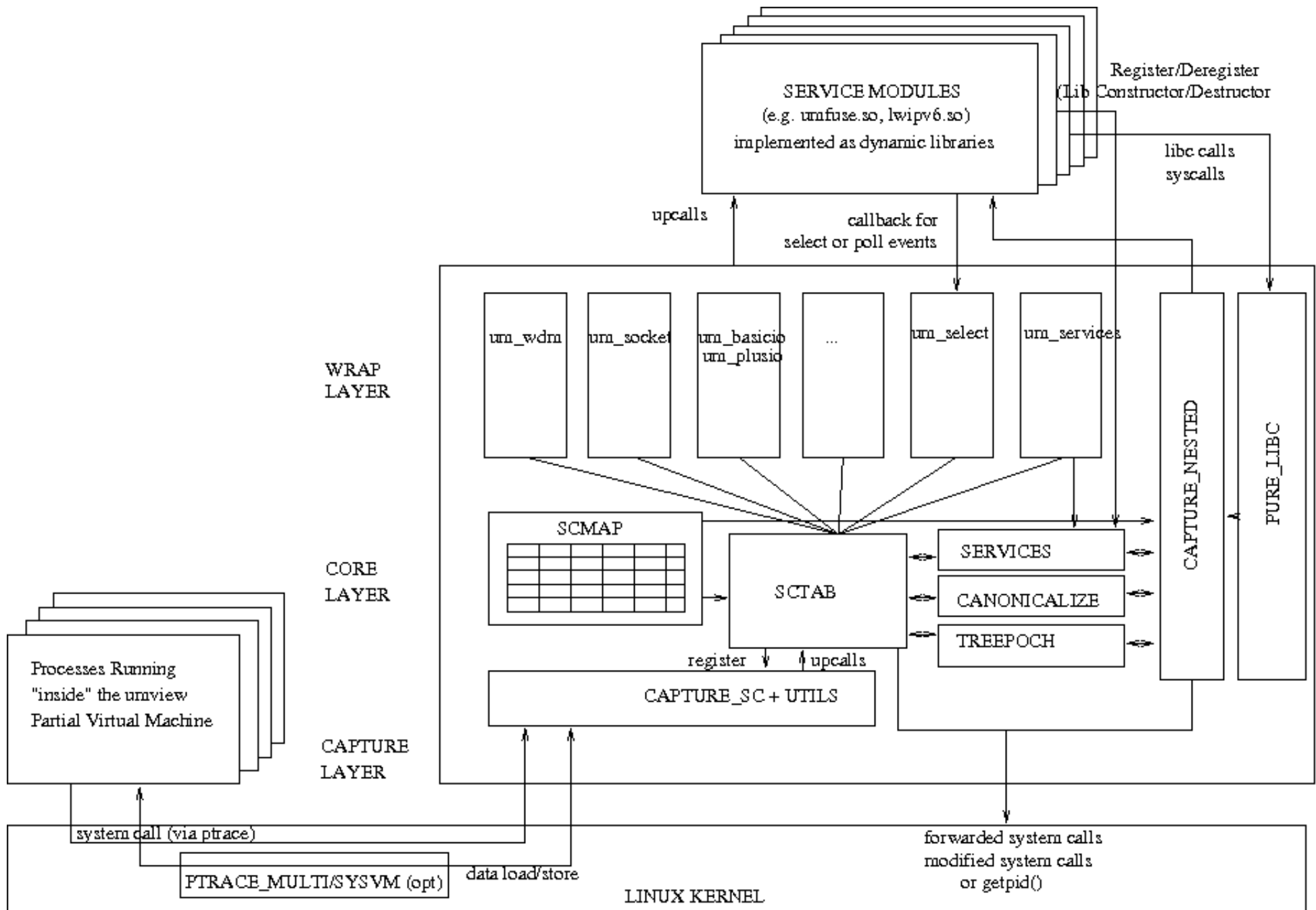


# Unfortunately...

- there are exceptions to the file-system naming scheme
- the most notably exception is networking:
  - network interfaces are devices but do not appear in /dev
  - there is one stack per protocol family in the kernel, no file system naming for stacks.
  - the “Berkeley socket” API does not (yet?) support multiple stacks.



# UMVIEW structure





# UMVIEW: capture of process syscalls

- UMVIEW uses ptrace to capture process syscalls. Ptrace was created for debuggers. The debugger (UMVIEW in our case) receives a signal each time one controlled process issues a system call, this latter process stops. The debugger can inspect and change the memory of the process and then restart it.
- It is the same technique used by user-mode linux.



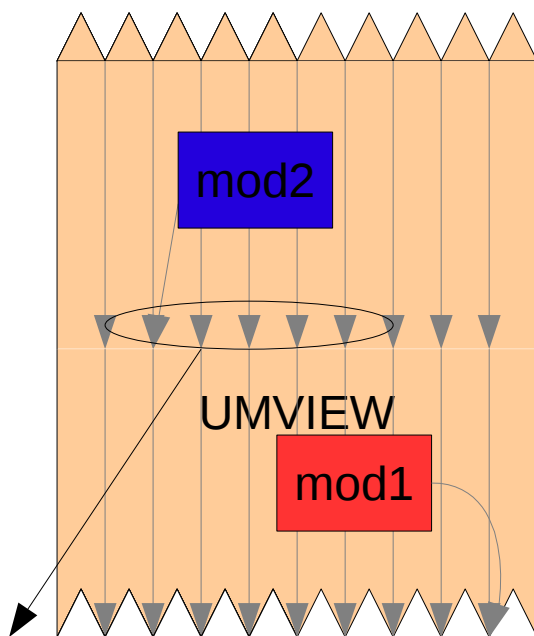
# UMVIEW: compatibility and performance

- ptrace is provided as a standard feature of the linux kernel
- UMVIEW runs linux unmodified binaries on any linux kernel (2.6)
- We provide patches for the kernel of the hosting system to increase the performance
  - PTRACE\_MULTI: executes several ptrace requests in a single call (including load/store of large chunks of data)
  - PTRACE\_SYSVM: provide a method to skip useless upcalls.
- ptrace was created for debugging. There is an international discussion about the best kernel interface for virtual machine support.



# Internal System Call Capture problem

- System calls generated by modules must be captured.
- Modules use libraries (that can do system calls!)
- Modules are loaded as shared libraries (for performance) thus run in the same addressing space
- There is the need for a “self-inspection” tool: a process should be able to capture the system call generated by itself!



Internal  
System Call  
Capture



# PURE\_LIBC

- Is an overlay library. It uses GLIBC and redefines some GLIBC calls. (e.g. syscalls, stdio)
- When loaded with LD\_PRELOAD, it captures the calls from the process involving system call.
- The interface is made by 4 lines of code:

```
typedef long int (*sfun)(long int __sysno, ...);  
extern sfun _pure_syscall;  
extern sfun _pure_socketcall;  
extern sfun _pure_native_syscall;
```

`_pure_syscall` and `_pure_socketcall` can be set to capture functions

`_pure_native_syscall` is the way to call the real syscall hidden by the library.



# UMVIEW modules

- A module implements one specific abstraction
  - umfuse: virtual file systems
  - umdev: virtual devices
  - lwipv6: (umlwip) virtual networking ....
- The interface for modules is composed by:
  - init: set a structure with all the managing functions
  - a “choice” function (the module asks to the question: “is this path/mounttype/protocol etc... up to you?”, returns a timestamp
  - one function for each managed system call, with the same interface of the system call.
  - one upcall function register for select/poll.



# UMFUSE

- It provides the virtual file system abstraction
- It needs submodules, user-mode implementations of the file system structures.
- It is a compatibility layer between UMVIEW and FUSE (M. Szeredi, [fuse.sf.net](http://fuse.sf.net), in the linux kernel since 2.6.15).
  - It is a source level compatibility, source code of FUSE modules must be compiled with different libraries to run with UMFUSE.



# UMFUSE FS modules

- UMFUSEEXT2(\*): ext2, ext3 file systems
  - uses the ext2fs lib
- UMFUSEISO9660(\*): iso9660 (rock\_ridge, joliet)
  - mounts also compressed iso images
  - uses the llibcdio/libiso9660 libs
- UMFUSEFAT(\*): fat FS
- UMFUSEENCFS: encoded FS
- UMFUSEFSSH: gives FS view to an ssh session
- UMFUSECRAM: cram (initrd), auto endianness(\*).
- UMFUSEFSFS(\*): Fast Secure File System (Cocchiaro, 2006), experimental

(\*): developed by the V<sup>2</sup> project, FUSE compatible.



# One word on FSFS

- Fast Secure File System is a secure network file system.
  - Encrypted NFS requires a lot processing power on the server (expecially with a large number of clients)
  - Users may want (need) their data to be stored in encrypted formats on hard disks (privacy disclosure for hard disk maintenance)
  - Solution: data is encrypted on disks, encrypted files are sent on a clear channel to the client. The cost of encryption is moved to client, data is stored and transmitted in an encrypted form.
- It is a bit more complex than this, control streams must be encrypted and some measures must be taken for “record&playback” attacks....



# UMFUSE @ work

- The use of umfuse is very simple.
  - note that the example uses the standard GNU-linux commands. (it is the same “mount” used by root)

```
xterm
renzo@titanic2:~/VIEWOS/UMVIEWOS$ um_add_service umfuse,so
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls -l /tmp/linux.img
-rw-r--r-- 1 renzo renzo 8388608 Aug 24 11:04 /tmp/linux.img
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls -l /tmp/f1
total 0
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umfuseext2 /tmp/linux.img /tmp/f1
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls -l /tmp/f1
total 23
drwxr-xr-x 2 root root 1024 Aug  9 2003 bin
drwxrwxr-x 2 root root 1024 Aug  9 2003 boot
drwxr-xr-x 2 root root 3072 Aug  9 2003 dev
drwxr-xr-x 2 root root 1024 Jan  9 2006 etc
drwxr-xr-x 2 root root 1024 Aug  5 2003 lib
drwxr-xr-x 2 root root 12288 Aug  5 2003 lost+found
drwxr-xr-x 4 root root 1024 Aug  5 2003 mnt
drwxr-xr-x 2 root root 1024 Aug  5 2003 proc
lrwxrwxrwx 1 root root  3 Aug  5 2003 sbin -> bin
drwxr-xr-x 4 root root 1024 Aug  5 2003 tmp
drwxr-xr-x 4 root root 1024 Aug  5 2003 usr
renzo@titanic2:~/VIEWOS/UMVIEWOS$ █
```



# UMFUSE @ hard work!

- In this example a file system mounted by ssh contains an encrypted fs. The encfs is mounted and there is an ext2 image inside. The third mount is on the same target dir of the first!

Passwords are entered in the UMVIEW console window.

```
xterm
renzo@titanic2:~/VIEWOS/UMVIEWOS$ um_add_service umfuse.so
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /tmp/enc /tmp/f1 /tmp/f2
/tmp/enc:
DlwcMrq0oia6f-HaNqnYYqFm okZ,G9ndcM1f90

/tmp/f1:

/tmp/f2:
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umfusessh localhost:/tmp/enc /tmp/f1
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /tmp/f1
DlwcMrq0oia6f-HaNqnYYqFm okZ,G9ndcM1f90
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umfuseencfs -o pre="" /tmp/f1 /tmp/f2
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /tmp/f2
ciao linux.img
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umfuseext2 /tmp/f2/linux.img /tmp/f1
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /tmp/f1
bin boot dev etc lib lost+found mnt proc sbin tmp usr
renzo@titanic2:~/VIEWOS/UMVIEWOS$
```





# UMDEV

- UMDEV implements virtual devices.
- The naming item for a device is a special file (some device can define several special files).
- From the process “view” access to devices is a standard file access (open/read/write/close...) + specific ioctl of the device.
- UMDEV is able to give the same “perception” to processes.



# UMVIEW and ioctl

- UMDEV provide ioctl virtualization
- ioctl is a “last resort” call able to manage hundreds of different requests with different arguments.
- ioctl itself is fd based so it is routed to the module who managed the open.
- there is a special call of the choice function (CHECKIOCTLPARMS) to define size/management/direction (input/output/I-O) of the args.



# UMDEV testmodules

- UMDEVNULL: test module. redefines a /dev/null device, echo what is sent to it
- UMTRIVHD: creates a ramdisk: a chunk of malloc-ed memory is given as a block device.
  - it is possible to create and mount a filesystem on it.



# UMDEVMBR

- This UMDEV submodule manages the MBR (EMBR) structure of disk images.
- When a disk image is mounted on a file (say /tmp/disk) it creates the special files to access all the partitions (/tmp/disk1 ... /tmp/disk63)
- It is possible to use fdisk (ddisk on powerpc linux) to repartition the hard disk, after the partition map reload, the new partitions are ready.



# UMDEV @ Work

- In this example:
  - I tried to partition my real HD /dev/hda as a user (permission denied)
  - I mounted a disk image on /dev/hda
  - In the new view I could change the partition table
  - I created an ext3 partition on /dev/hda5
  - I mounted the new partition on /mnt (umfuseext2)

```
xterm
renzo@titanic2:~/VIEWOS/UMVIEWOS$ um_add_service umdev.so
renzo@titanic2:~/VIEWOS/UMVIEWOS$ /sbin/ddisk /dev/hda

Unable to open /dev/hda
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls -l /tmp/hd10
-rw-r--r-- 1 renzo renzo 10485760 Aug 24 11:50 /tmp/hd10
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umdevmbr /tmp/hd10 /dev/hda
renzo@titanic2:~/VIEWOS/UMVIEWOS$ /sbin/ddisk /dev/hda

Command (m for help): p

Disk /dev/hda: 10 MB, 10485760 bytes
16 heads, 16 sectors/track, 80 cylinders
Units = cylinders of 256 * 512 = 131072 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1            1           10        1272    83   Linux
/dev/hda2            11           80        8960     5   Extended
/dev/hda5            11           80        8952    83   Linux

Command (m for help): q

renzo@titanic2:~/VIEWOS/UMVIEWOS$ /sbin/mkfs.ext3 /dev/hda5
mke2fs 1.39-WIP (10-Dec-2005)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
2240 inodes, 8952 blocks
447 blocks (4.99%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
1120 inodes per group
Superblock backups stored on blocks:
    8193

Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 23 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
renzo@titanic2:~/VIEWOS/UMVIEWOS$ um_add_service umfuse.so
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /mnt
renzo@titanic2:~/VIEWOS/UMVIEWOS$ mount -t umfuseext2 /dev/hda5 /mnt
renzo@titanic2:~/VIEWOS/UMVIEWOS$ ls /mnt
lost+found
renzo@titanic2:~/VIEWOS/UMVIEWOS$ umount /mnt
renzo@titanic2:~/VIEWOS/UMVIEWOS$ umount /dev/hda
renzo@titanic2:~/VIEWOS/UMVIEWOS$ █
```



# UMDEVTAP

- Virtual TAP
- emulates the standard tuntap interface (/dev/net/tun)
- Net packets get sent to a vde switch



# UMBINFMT

- It implements the same interface of the BINFMT kernel module.
- It is possible to choose the “interpreter” for an executable depending on file extensions or magic numbers (pattern matching).



# UMBINFMT @ Work

In this example

- on a linuxppc I show two executables for other architectures
- I load Aumbinfmt
- I mount it on the same dir of the kernel one (can be mounted elsewhere, scripts are compatible if the same dir is used)
- I register the interpreter (QEMU)
- The executables run (ls i386 is a real ls, ls on arm is busybox command)

```
xterm
renzo@titanic2:~/VIEWOS/UMVIEWOS$ cd ~/tests/qemu/qemu-tests
renzo@titanic2:~/tests/qemu/qemu-tests$ file i386/ls
i386/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux
2.0.0, dynamically linked (uses shared libs), stripped
renzo@titanic2:~/tests/qemu/qemu-tests$ file arm/ls
arm/ls: ELF 32-bit LSB executable, ARM, version 1 (ARM), for GNU/Linux 2.2.0, dy
namicly linked (uses shared libs), stripped
renzo@titanic2:~/tests/qemu/qemu-tests$ uname -a
Linux titanic2 2.6.15-viewos #3 PREEMPT Sun Feb 5 18:16:30 CET 2006 ppc GNU/Linu
x
renzo@titanic2:~/tests/qemu/qemu-tests$ ls
arm i386 linux ppc sparc
renzo@titanic2:~/tests/qemu/qemu-tests$ ./i386/ls
bash: ./i386/ls: cannot execute binary file
renzo@titanic2:~/tests/qemu/qemu-tests$ ls -l /proc/sys/fs/binfmt_misc
total 0
--w----- 1 root root 0 Aug  2 10:23 register
-rw-r--r-- 1 root root 0 Aug  2 10:23 status
renzo@titanic2:~/tests/qemu/qemu-tests$ um_add_service umbinfmt,so
renzo@titanic2:~/tests/qemu/qemu-tests$ mount -t umbinfmt none /proc/sys/fs/binf
mt_misc
renzo@titanic2:~/tests/qemu/qemu-tests$ ls -l /proc/sys/fs/binfmt_misc
total 0
--w----- 1 root root 0 Jan  1  1970 register
-rw-r--r-- 1 root root 0 Jan  1  1970 status
renzo@titanic2:~/tests/qemu/qemu-tests$ echo ':i386:M::\x7fELF\x01\x01\x01\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x03\x00;\xff\xff\xff\xff\xff\xfe\xfe\xff\
\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/local/bin/qemu-i386:' > /pr
oc/sys/fs/binfmt_misc/register
renzo@titanic2:~/tests/qemu/qemu-tests$ ./i386/ls
arm i386 linux ppc sparc
renzo@titanic2:~/tests/qemu/qemu-tests$ ./arm/ls
bash: ./arm/ls: cannot execute binary file
renzo@titanic2:~/tests/qemu/qemu-tests$ echo ':arm:M::\x7fELF\x01\x01\x01\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x28\x00;\xff\xff\xff\xff\xff\xff\xff\x0
0\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/local/bin/qemu-arm:' > /p
roc/sys/fs/binfmt_misc/register
renzo@titanic2:~/tests/qemu/qemu-tests$ ./arm/ls
arm
i386
linux
ppc
sparc
renzo@titanic2:~/tests/qemu/qemu-tests$ █
```



# UMLWIP

- It is the virtual networking module
- UMLWIP uses LWIPV6
- The module to load is named lwipv6.so
- It creates virtual interfaces.
- Compatible with the modern iputils.
- After the module has been loaded the interfaces can be configured and used.



# UMLWIP @ Work

- lwipv6.so without extra options creates two interfaces lo0 and vd0 (vde)
- vd0 is configured using iputils (Ipv6 gets auto-configuration)
- ssh to a remote host shows we are using the virtual network

```
xterm
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ um_add_service lwipv6.so
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ip link
1: lo0: <LOOPBACK,UP> mtu 0
   link/loopback
2: vd0: <BROADCAST> mtu 1500
   link/ether 02:02:6e:94:82:06 brd ff:ff:ff:ff:ff:ff
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ip addr add 130.136.31.22/24 dev
vd0
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ip route add default via 130.136
.31.1
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ip link set vd0 up
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ip addr
1: lo0: <LOOPBACK,UP> mtu 0
   link/loopback
   inet6 ::1/128 scope host
   inet 127.0.0.1/8 scope host
2: vd0: <BROADCAST,UP> mtu 1500
   link/ether 02:02:6e:94:82:06 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::2:6eff:fe94:8206/64 scope link
   inet 130.136.31.22/24 scope global
   inet6 2001:760:2e00:ff00:2:6eff:fe94:8206/64 scope global
renzo@titanic2:~/VIEWOS/UMVIEWOS/um_testmodule$ ssh sockmel.bononia.it
fcntl(6, F_GETFL, 0): Function not implemented
Linux sockmel 2.6.16-sockmel #1 Tue May 9 10:04:42 CEST 2006 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

You have new mail.

Last login: Thu Aug 24 14:35:14 2006 from 130.136.31.22
renzo@sockmel:~$ w
 14:36:00 up 107 days,  3:55,  1 user,  load average: 0.08, 0.04, 0.01
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
renzo     pts/0    130.136.31.22  14:35    0.00s  0.02s  0.01s  w
renzo@sockmel:~$ █
```



# VIEWFS

## (under development)

- ViewFS is a File system remapping module
- Some features:
  - Directory merge,
  - WORM access,
  - file and directory hiding



# UMMISC

- This module virtualize several concepts like:
  - process ids
  - time
  - system id
  - user/group id



# Kmview: Kernel Module View-OS

- Based on the utrace kernel (Roland McGrath)
- Very efficient and transparent capture.
- Several optimizations added:
  - fd based selection
  - magicpoll



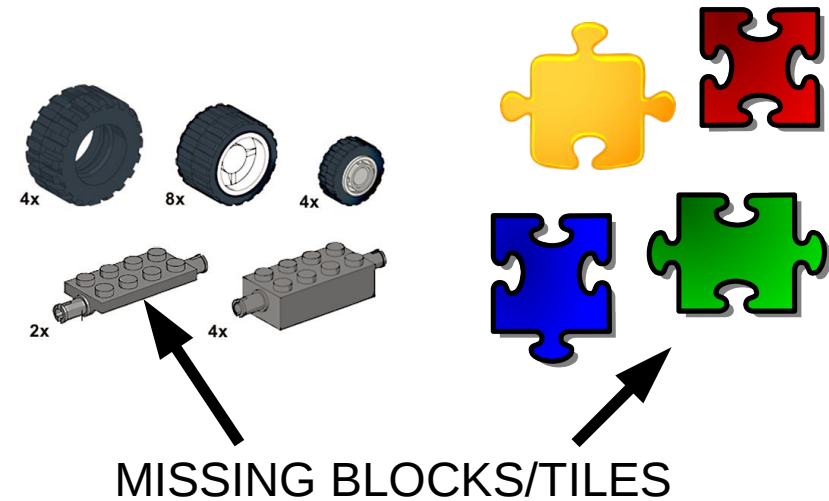
# Virtual Square usage scenarios

- .... are so many that this should be a specific seminar, not a slide.
- Prototyping
- Security
- in general “user freedom” ;-)
- Education: Virtual Square can be used for lab exercises in System Architecture, Operating Systems, Networking, Security, Distributed Systems etc...
- Virtual Square in Education: is the topic of an international cooperation with Prof. Michael Goldweber, Xavier Univ. Cincinnati, OH.



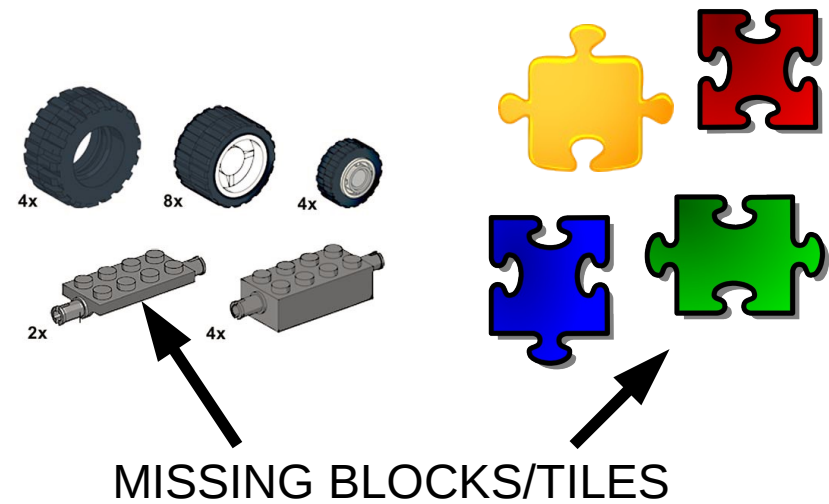
# Virtual Square: what is next?

- LWIPv6 internal dhcp client
- VDE-XEN gateway
- SNMP support for VDE
- Mobility support for VDE
- UMVIEW Modules:
  - Viewfs
  - RemoteSyscall
  - Network multistack (recursive LWIP)
- Extension to the model
  - consistent view of virtual users





# Still Missing...

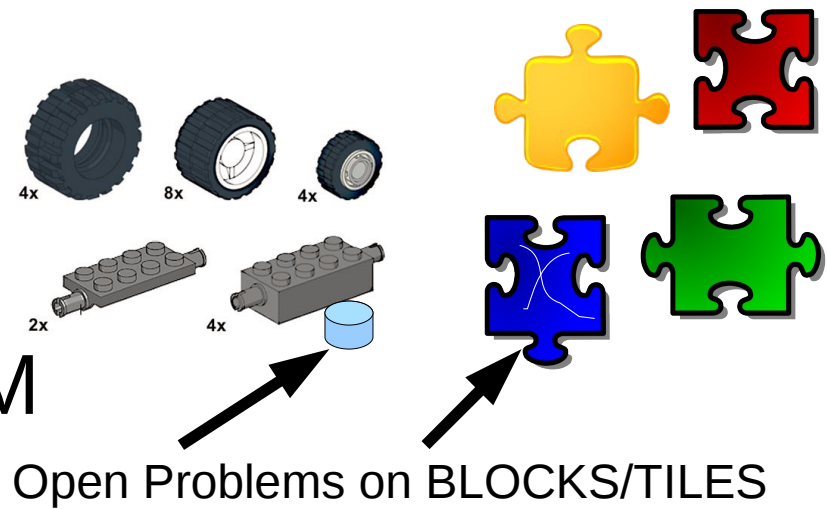


- Support for Klik filesystem
- Cross Platform local Procedure Call.
- Virtual SysV IPC
- WWFX
- Porting to other OS (MacOSX)
- VDE SlirpV6



# Open Problems...

- Native Kernel support for VM
- mmap (write access)
- Museum (licenses!)
- Bugs... (there is always one around the corner)





# Kernel, $\mu$ Kernel and mKernel!

- (Monolithic) Kernels
  - bad design, efficient
  - Linux is very well supported and widely applied
- Micro Kernels
  - nice design, inefficient
  - Lack of drivers, rarely applied outside research
    - (or used with monolithic kernels loaded as servers)
- MilliKernel (NEW!)
  - modular monolithic kernel able to use external servers for hi-level structures (File System, Networking, Memory Management Policies) or for lo-level structures (drivers).
  - It is up to the user/administrator to decide what should be load as server or inside the kernel (efficiency vs. reliability+flexibility).



# Idea...

VIEW-OS/UMVIEW modules seems to be modules for a Linux Millikernel structure...



# The Operating System ZOO

[www.oszoo.org](http://www.oszoo.org)



# OSZOO

- OSZOO is a repository of Disk Images for VM.
- It is possible to test an O.S. just by downloading the image and run it.
- html and bittorrent access to images.

The screenshot shows the main page of the FreeOsZoo project. The browser window title is "Main Page - FreeOsZoo - Firefox". The address bar shows the URL "http://www.oszoo.org/wiki/index.php/Main\_Page". The page content includes a navigation menu on the left with links like "Main Page", "Community portal", and "Recent changes". The main content area features a "Contents" list with 11 items, a "Welcome to the FreeOsZoo Project!" section, and an "About the FreeOsZoo" section. The "Welcome" section states: "With FreeOsZoo, the dream of running several Operating Systems on a single computer comes true. FreeOsZoo provides ready-to-run images of QEMU virtual computers, pre-installed with a Free Operating System and a set of popular free software. To get started, you only need to install QEMU and download a single file from the FreeOsZoo project. This wiki page is going to be the main reference of the project but it's still work in progress. Help us or just don't complain! :-)"



# LIVE OSZOO

- Now it is possible to run OS images directly on our servers.
- Users need just a browser with jvm support.

Free Live OS Zoo - Firefox

File Edit View Go Bookmarks Tools Help

http://connessi.webminds.cs.unibo... Go

The Mozilla Organi... Latest Builds VOGLIO BREVETTA... Corso di Sistemi O...

## Free Live OS Zoo

FreeOsZoo Home

Available displays: 15/15

**IMPORTANT: This is EXPERIMENTAL. Before use, read carefully ALL of [this page](#).**

-----

**Select Image to load:**

- CentOS GNU/Linux 4 (800x600, keymap is it)
- Fedora GNU/Linux Core 5 (800x600)
- FreeBSD 6.0
- FreeDOS b9r5 (no login required)
- Minix 3.1.1 (root passwd not set)
- OpenSolaris + Slackware GNU/Linux (root passwd is root for Solaris)
- Plan 9 release 4 (1024x768, user glenda, no passwd)
- ReactOS 0.2.7 (no passwd required)
- Ubuntu GNU/Linux 6.06 LTS (800x600)
- Gentoo GNU/Linux 2006.0

The zoo uses a Java applet.

Select your ACTUAL keyboard mapping:

In order for your keyboard to behave properly during emulation, you MUST choose your actual keyboard mapping.

If you want to use a different keymap in the emulated OS, choose it there.

Unless otherwise specified, all images are set for us keyboard and have a root user with passwd "piripicchio" and applet size will be 725x485

Done



# Fedora5 on Live OSZOO

This is the  
example of a  
GNU-Linux  
Fedora Core 5  
loaded on a  
Live OSZOO  
virtual  
machine.

## Free Live OS Zoo

[FreeOsZoo Home](#)

Running image "Fedora GNU/Linux Core 5 (800x600)" in QEMU/VNC display :1  
Unless otherwise specified, the image started with a us keymap and a root user with passwd "piripicchio"

If after a change of resolution the screen is unclear, reload the page.  
To use keyboard in the emulated OS, click on its screen.

When you're done using this, you can close the browser window or shut down the image using this button: [Shutdown](#)

**This image will be killed @ 2006-08-24 16:59:33 CEST**





# The Virtual Square TEAM

- Renzo Davoli – rd235, reenzo (designer, main developer)
- Ludovico Gardenghi – garden (viewfs, optimization, umview inter module communication, logging interface)
- Michael Goldweber, Mauro Morsiani (MPS, uMPS)
- Filippo Giunchedi – godog (Debian maintainer, vde-snmp)
- Luca Bigliardi – shammash (libcomm, autolink)
- Diego Billi (lwipv6 filtering, optimization)
- Andrea Gasparini – gaspa (nesting, kernel-patches)
- Daniele Lacamera – danielinux (vde\_cryptcab, debian packets, vde-l3).
- Mattia Gentilini MG55 (freelive oszoo)
- Andrea Forni - Gendag (remote procedure call, Slirpv6)
- Andrea Seraghiti, Mattia Belletti,..... and many others.



renzo@cs.unibo.it

To be continued...