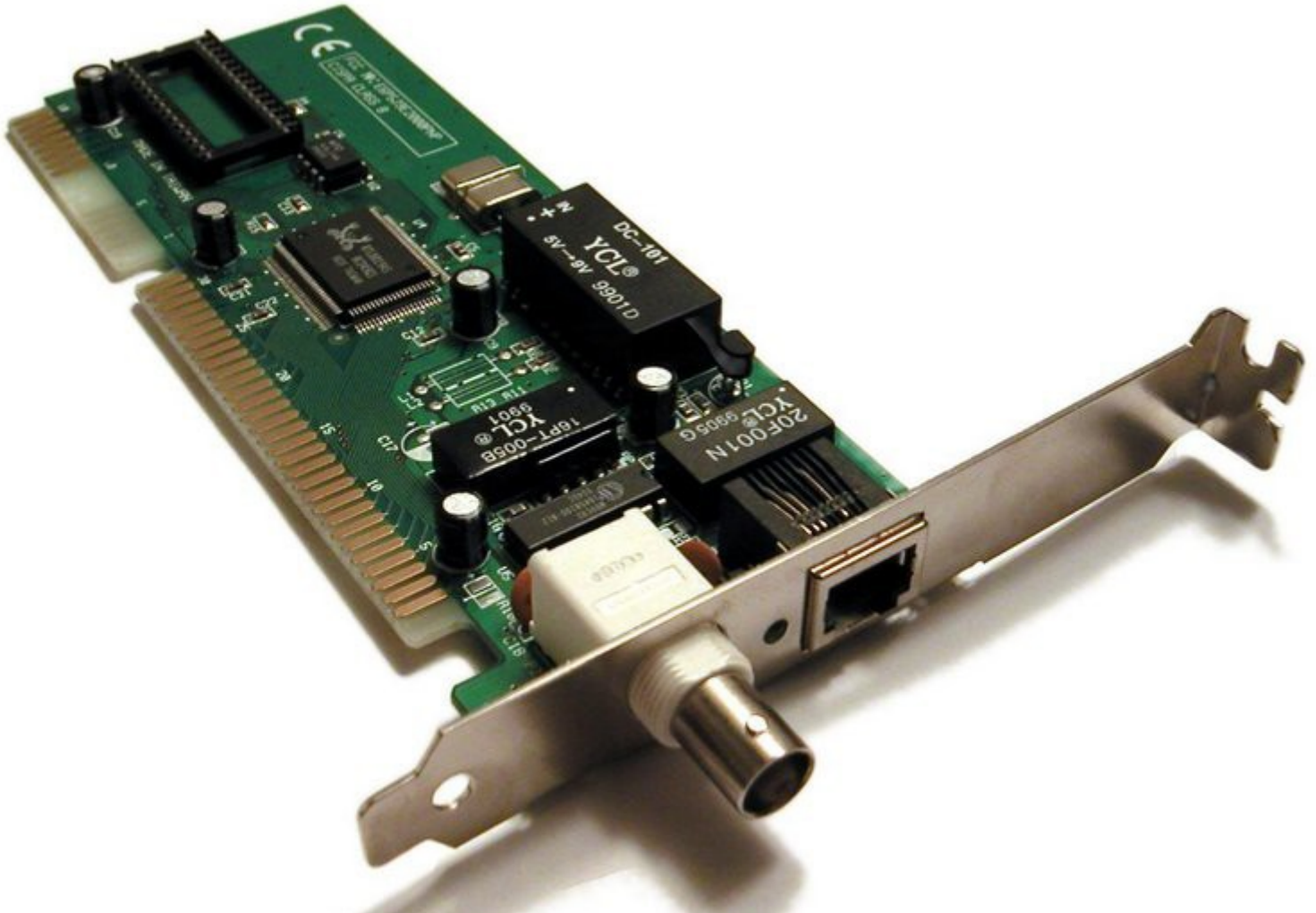
Internet of Threads

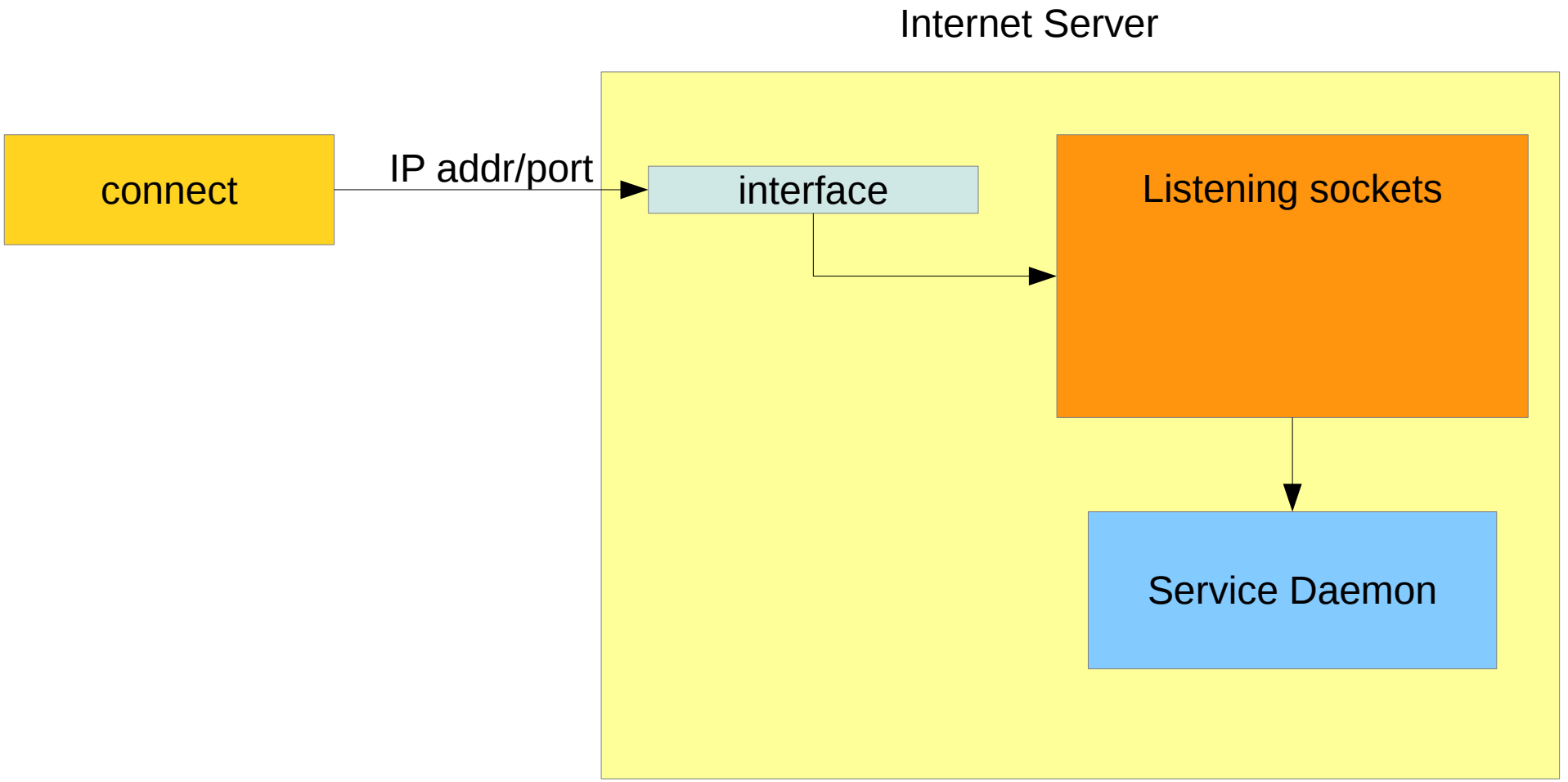
Renzo Davoli
Dipartimento di Scienze dell'Informazione
Universita' di Bologna

Quali sono i nodi di Internet?

- L'entita' indirizzabile per IP e' l'interfaccia.



Visione “storica”



Oggi: nodi “virtuali” di rete

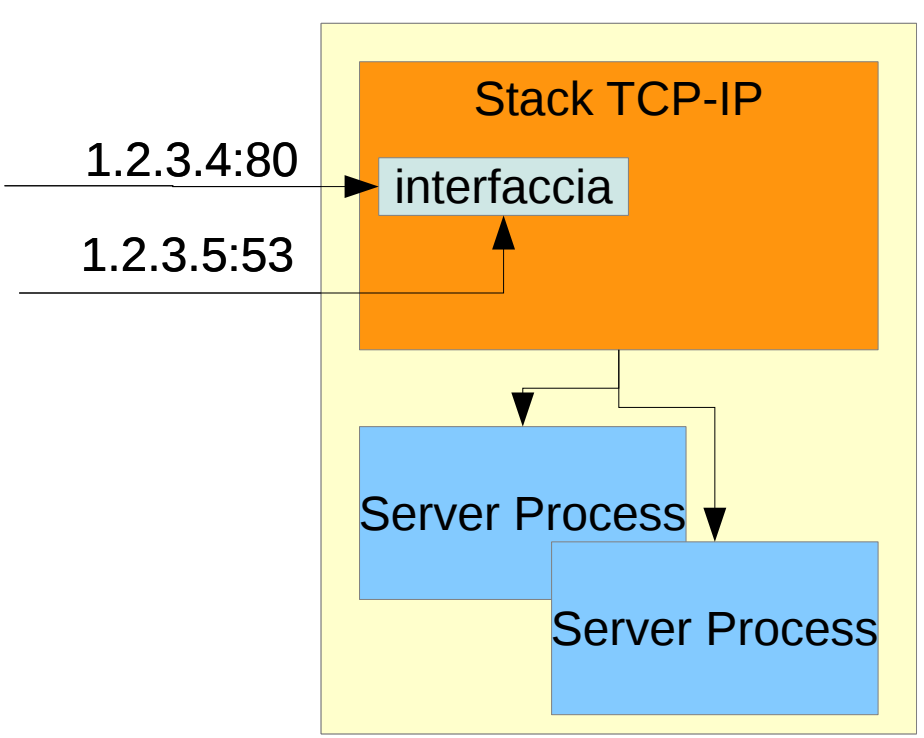
- Indirizzi associati a macchine virtuali (xen, kvm, user-mode linux etc)
- Indirizzi multipli per server ad alta affidabilita' (associati a servizi, per la migrazione in caso di fault).

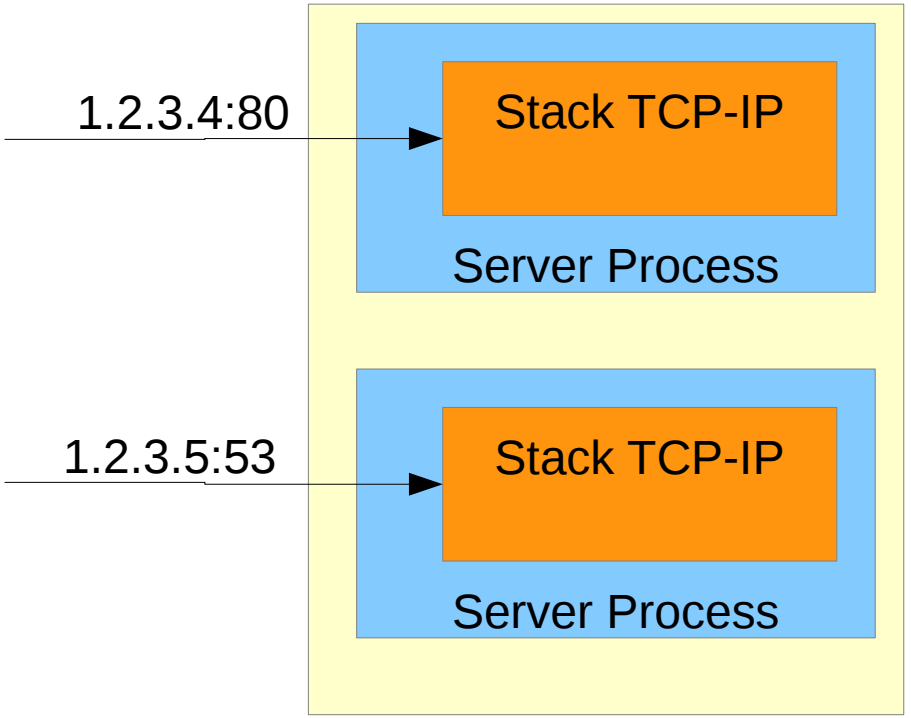
Esempi di Problemi “difficili” in un sistema multiuser/multitasking

- Assegnare IP diversi agli utenti che lavorano contemporaneamente
- Assegnare QoS/routing differenti a processi in esecuzione
- Avere piu' server attivi per lo stesso servizio (stesse porte)
- Poter usare una VPN per alcuni processi e la rete reale per altri.
- Migrare un processo su un altro sistema mantenendo le connessioni attive.

Internet of Threads

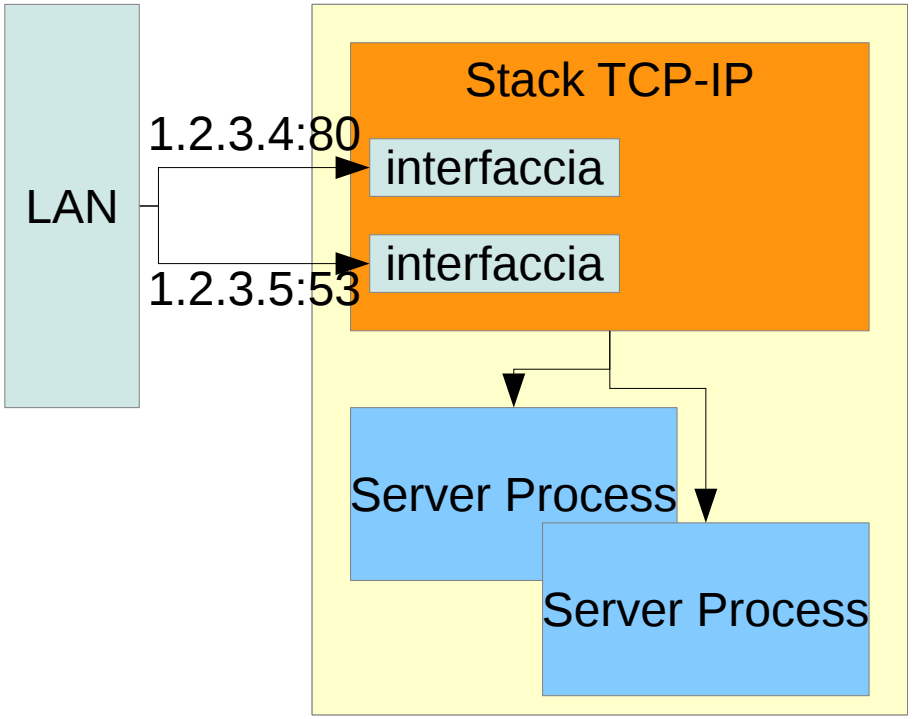
- I nodi della rete devono diventare (anche) i processi che forniscono i servizi.

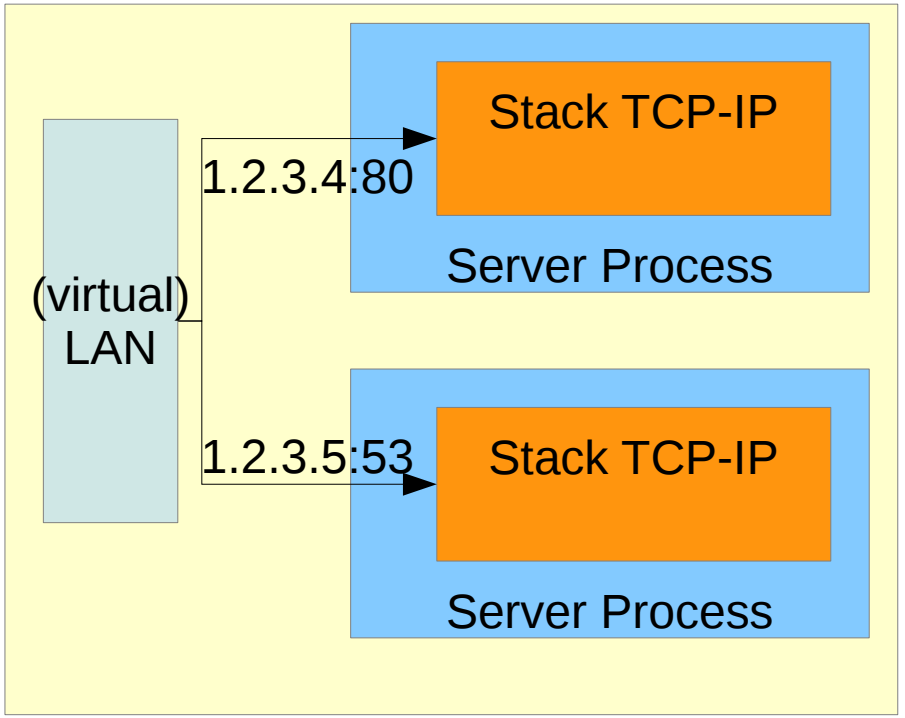




Internet of Threads
where is the LAN?

- La rete locale che connette i processi deve essere virtuale...





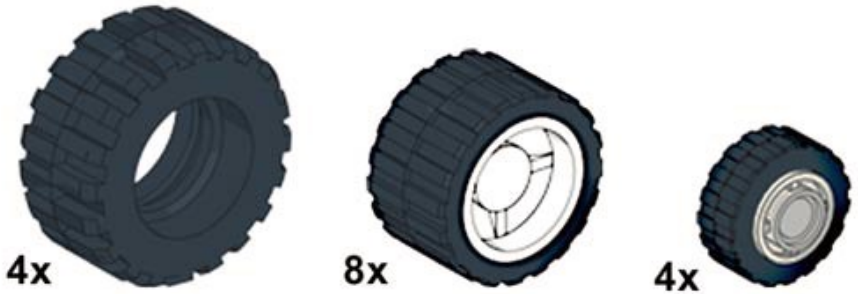
Virtual Ethernet

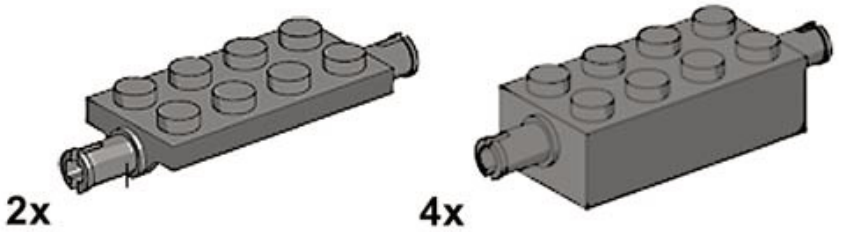
- I processi si collegano fra loro e a Internet usando reti virtuali.
- Gli stessi strumenti oggi usati nelle reti locali per collegare personal computer possono venir utilizzati nelle reti virtuali per interconnettere processi. (e.g. un processo puo' usare dhcp).
- Si possono creare reti locali ibride, in parte reali in parte virtuali.

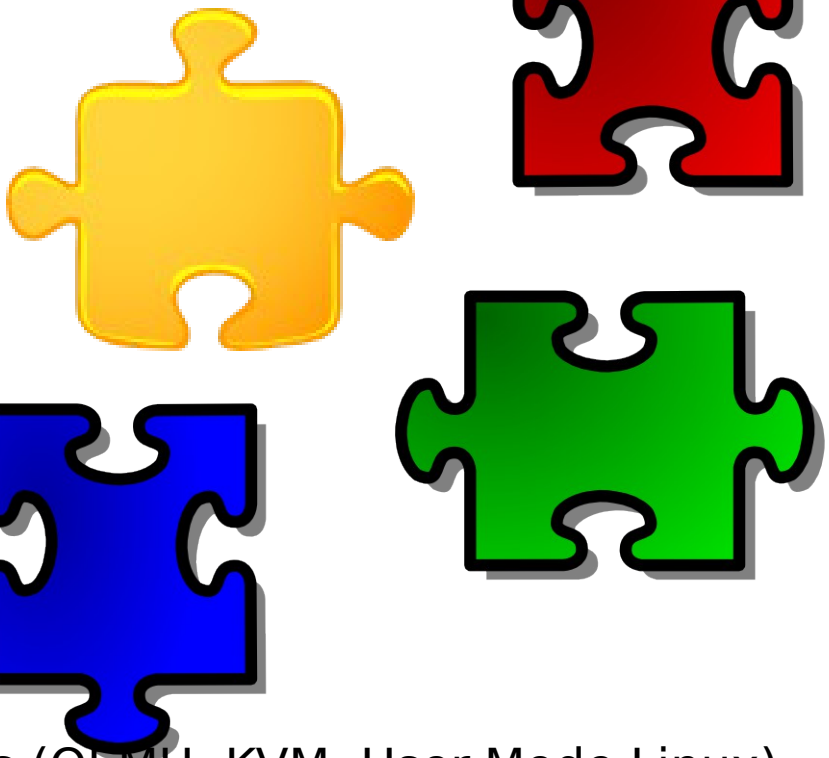
VIRTUAL SQUARE LAB

- An international lab on virtuality
- Project repository:
  - Virtual Distributed Ethernet (VDE)
  - LWIPv6
  - PureLibC
  - View-OS
    - umview/kmview
    - modules

Virtual Square







Virtual Machines (QEMU, KVM, User-Mode Linux),
Virtual Networks (e.g. Overlay networks, VPNs),
Virtual Execution Environments,
Virtual File systems,
Virtual root (chroot),
Virtual users(e.g. fakeroot),
Virtual Time,
Virtual Honey Pots,
Virtual executable intepreters (e.g. binfmt), ...

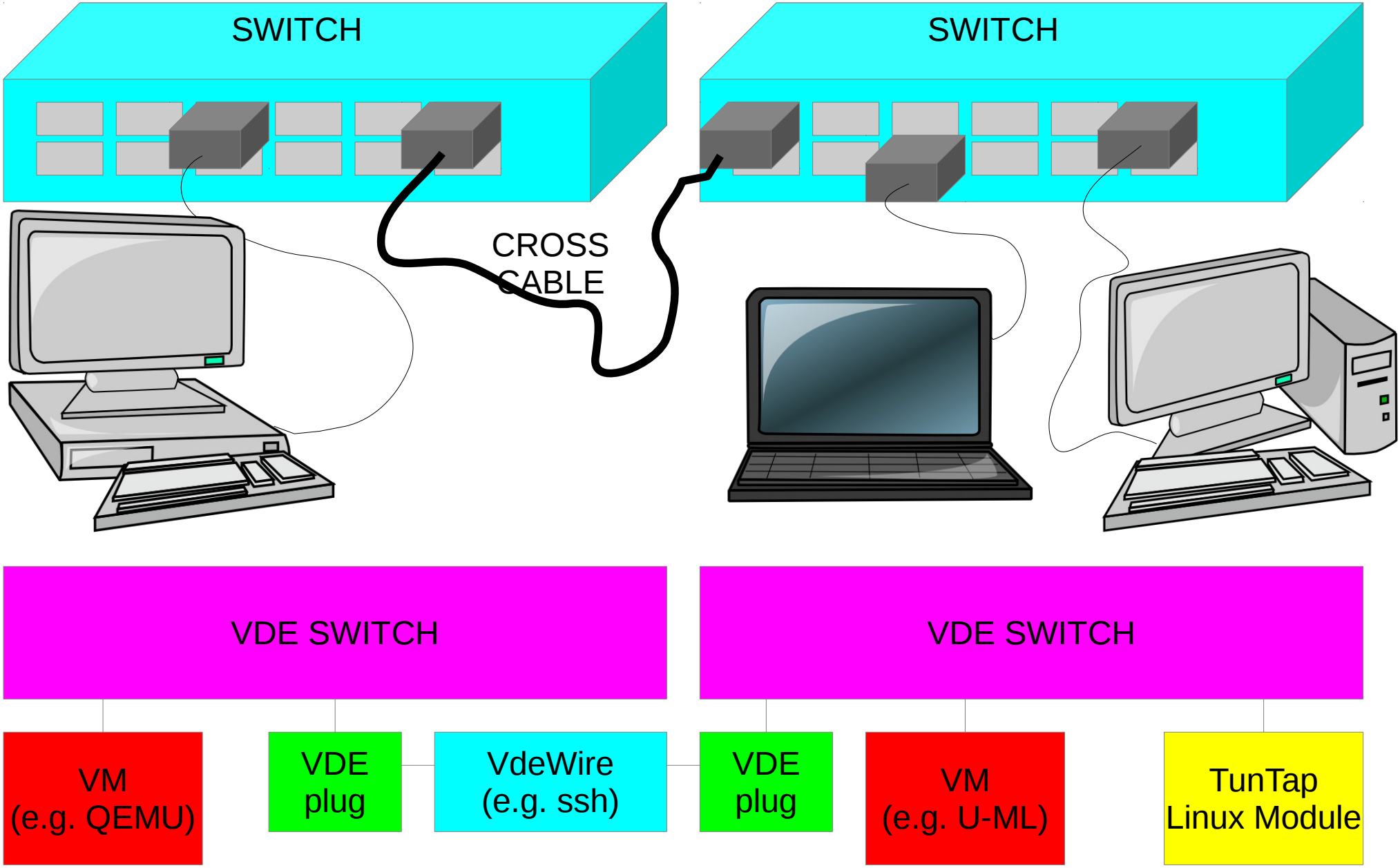
Virtual Square Goals

- Communication
  - Different *virtualities* must be interconnected.
- Integration
  - Seeing how specific *virtualities* can be seen as special cases of broader ideas of *virtuality.*
- Extension
  - Several needs could be captured by some model of *virtuality;* V^2 seeks a unifying idea of *virtuality* able to provide a consistent and extensible view.

Virtual Square does it better!

- Virtual Square ha sviluppato gli strumenti per poter sperimentare fin da ora sull'Internet of Threads:
  - VDE: virtual distributed ethernet
  - LWIPv6: an entire IPv4/v6 hybrid stack as a library
  - View-OS: view-based operating system.
    - Umview/kmview: View-OS implementations as partial virtual machines running on GNU-Linux.
- E' un proof-of-concept usabile!

Virtual Distributed Ethernet (VDE)



VDEv2: advertisement

- VDEv2:
  - modular design
  - compatible with user-mode linux, qemu, kvm, virtualbox, 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!)

LWIPv6

- It is an IPv4/v6 stack implemented as a library.
- Fork project from LWIP project (Adam Dunkels <adam@sics.se>)
- Can be connected to any number of VDE, TUN, TAP, SLIRP 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
- NATv4 NATv6
- DHCP server/client, RADV server

Berkeley sockets API: problem #1

- The Berkeley Sockets API has been designed for **one** protocol stack (per protocol family).
  - Multiple stacks => different networking features (per user, per application...)
- Unix uses the file system as a naming space for everything (devices, kernel variables, ...) **except for networking**.
  - Access control to networking

Solution #1: msockets

```
#include <msocket.h>

int msocket(char *path, int domain, int type, int protocol);
```

- Path is the pathname of the stack
- domain/type/protocol are the same defined in socket(2).
- A stack is a special file (new type of special file, see stat(2)):

- Each process has a default stack for each protocol family (domain).
  - If **`path==NULL`**, msocket uses the default stack.
- It is backwards compatible.

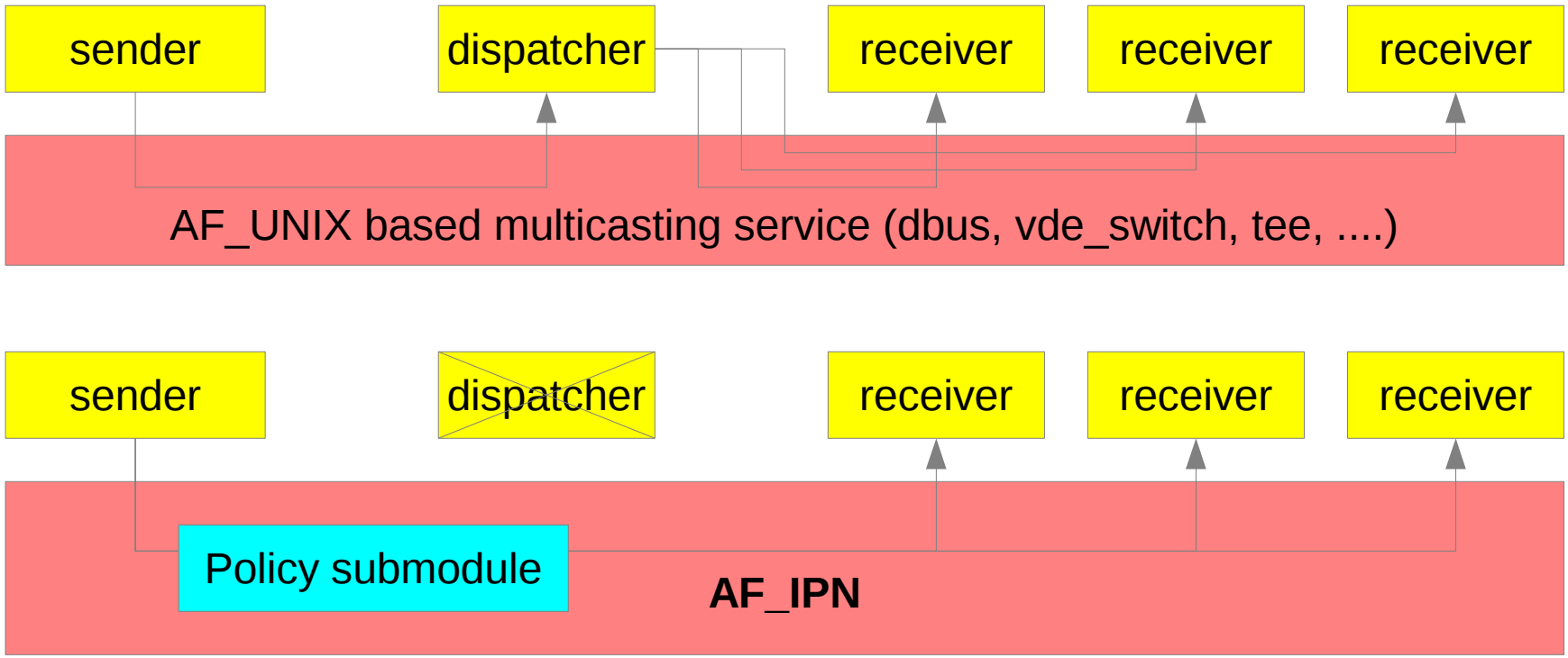
Berkeley sockets API: problem #2

- Berkeley Sockets API does provide support for IPC (AF_UNIX)
- Berkeley Sockets API does not provide support for multicast IPC
- Berkeley Sockets is mainly for point-to-point, client-server communication

- Many applications need multicast IPC (dbus, vde_switch, midi-patchbay, mpeg-ts demultiplexing...)

IPN=Inter Process Networking

- IPN is for IPC (like AF_UNIX)
- IPN provides **fast, kernel implemented,** multicast communication among processes.



IPN: implementation

- A new address family AF_IPN
- Policies can be provided as submodules.
  - IPN_BROADCAST (default) each message is delivered to all the members but the sender
  - IPN_VDESWITCH a virtual ethernet switch
  - IPN_MPEGTS mpeg transport stream demultiplexing
- Two services (sockopt selectable):
  - LOSSLESS: bounded buffer approach, late receivers delay senders
  - LOSSY: late receivers lose data.

View-OS

... a process with a view

Each process should be permitted to have its own ***view*** of the execution environment

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
  - ...

HANDS ON!
How to start a View-OS monitor:

```
user@host:~$ umview bash
This kernel supports: PTRACE_MULTI PTRACE_SYSVM ppoll
View-OS will use: PTRACE_MULTI PTRACE_SYSVM ppoll

pure_libc library found: syscall tracing allowed

rd235 2.6.29-utrace GNU/Linux/View-OS 10585 0
user@host[10585:0]:~$
```

- Umview runs on vanilla Linux kernels, Kmview requires a kernel module loaded (and utrace).
- Instead of bash one may run his/her favorite executable (e.g. xterm, script....)

Virtual Networking by View-OS

```
$ um_add_service umnet
$ mount -t umnetlwipv6 none /dev/net/default
$ ip link set vd0 up
$ ip addr add 10.1.2.3/24 dev vd0
$ 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 02:02:5a:44:e2:06 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::2:5aff:fe44:e206/64 scope link
    inet 10.1.2.3/24 scope global
```

- A network stack can be “mounted.”
- /dev/net/default is the default stack, but View-OS supports multiple stacks.

Virtual Networking

```
$ um_add_service umnet
$ mount -t umnetlwipv6 -o tn0=tunx none /dev/lwip0
$ mount -t umnetlwipv6 -o tp0=tapx,vd0=/tmp/switch none /dev/lwip1
$ mstack /dev/lwip0 ip addr
1: lo0: <LOOPBACK,UP> mtu 0
    link/loopback
    inet6 ::1/128 scope host
    inet 127.0.0.1/8 scope host
2: tn0: <> mtu 0
    link/generic
$ mstack /dev/lwip1 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> mtu 1500
    link/ether 02:02:47:98:ad:06 brd ff:ff:ff:ff:ff:ff
3: tp0: <BROADCAST> mtu 1500
    link/ether 02:02:03:04:05:06 brd ff:ff:ff:ff:ff:ff
$
```

Scenari di utilizzo:

- Servizi su reti diverse:
  - Piu' browser attivi con “prospettive” diverse
- Virtual appliance/virtual embedded appliance
  - NAS virtuali
  - accesso web alle applicazioni dell'utente

I problemi “difficili” diventano “facili”

- Assegnare IP diversi agli utenti che lavorano contemporaneamente
- Assegnare QoS/routing differenti a processi in esecuzione
- Avere piu' server attivi per lo stesso servizio (stesse porte)
- Poter usare una VPN per alcuni processi e la rete reale per altri.
- Migrare un processo su un altro sistema mantenendo le connessioni attive.

Una parola sulla sicurezza

- PRINCIPIO DEL PRIVILEGIO MINIMO
  - Ogni applicazione deve accedere alle sole risorse necessarie per l'elaborazione richiesta
- Con Internet of threads:
  - Non c'e' necessita' di privilegi di amministrazione (che possono mettere a rischio l'infrastruttura)
  - La separazione degli stack riduce i rischi di effetto domino in caso di servizi “fragili”.

Una parola sulle prestazioni

- La presenza di molteplici (numerosi) stack richiede maggiore memoria
- La rete virtuale puo' smistare pacchetti a livello kernel (gia' implementato in IPN)
- Il codice puo' essere condiviso e gli stack possono diventare librerie kernel

I prossimi 20 anni...

- La rete e' e restera' un grande laboratorio di comunicazione.
- La ricerca non riguarda solo i servizi, anche la struttura stessa della rete puo' essere reinventata.