FOSDEM

FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

FOSDEM 2012

Internet of Threads

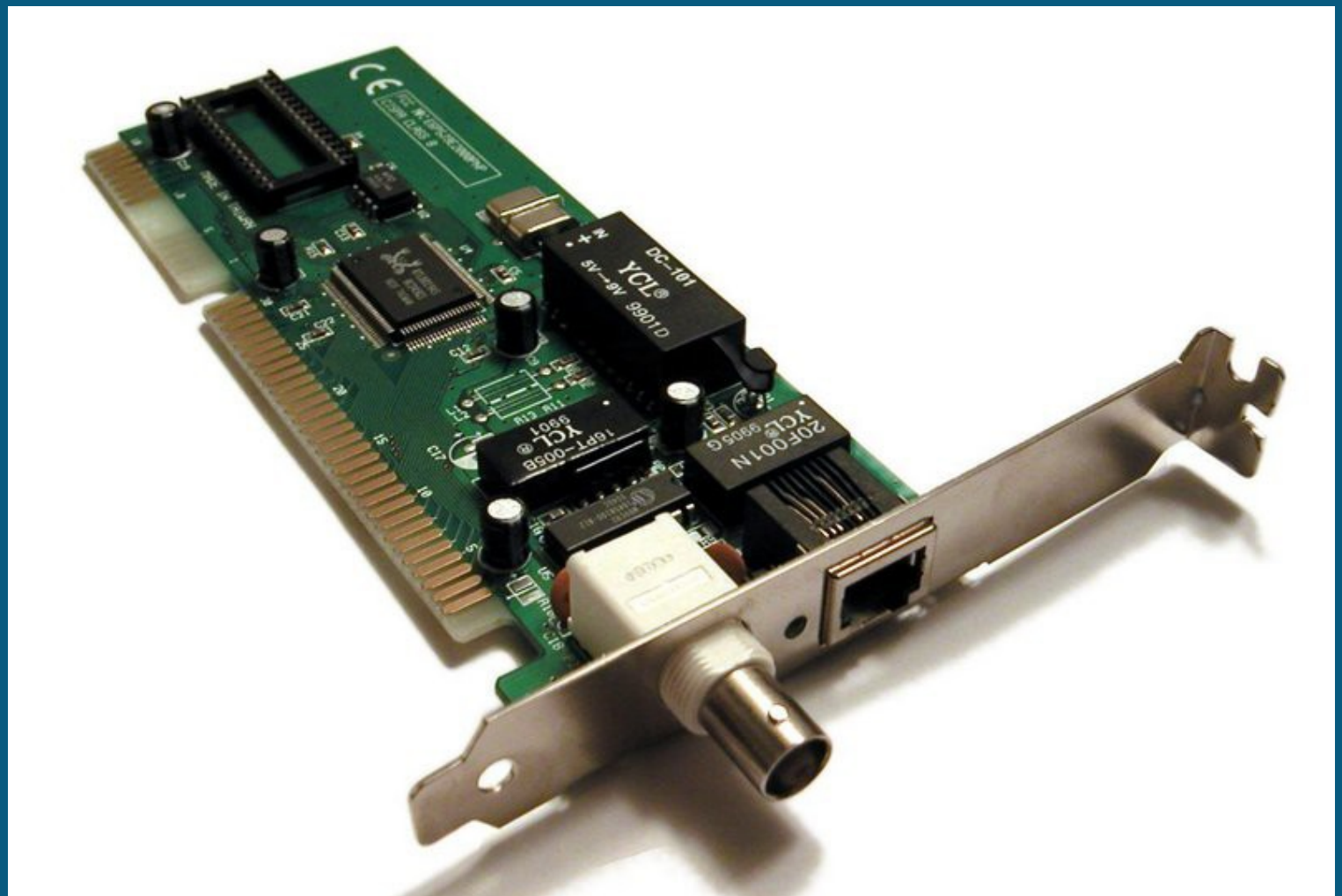
Renzo Davoli

Virtual Square Lab

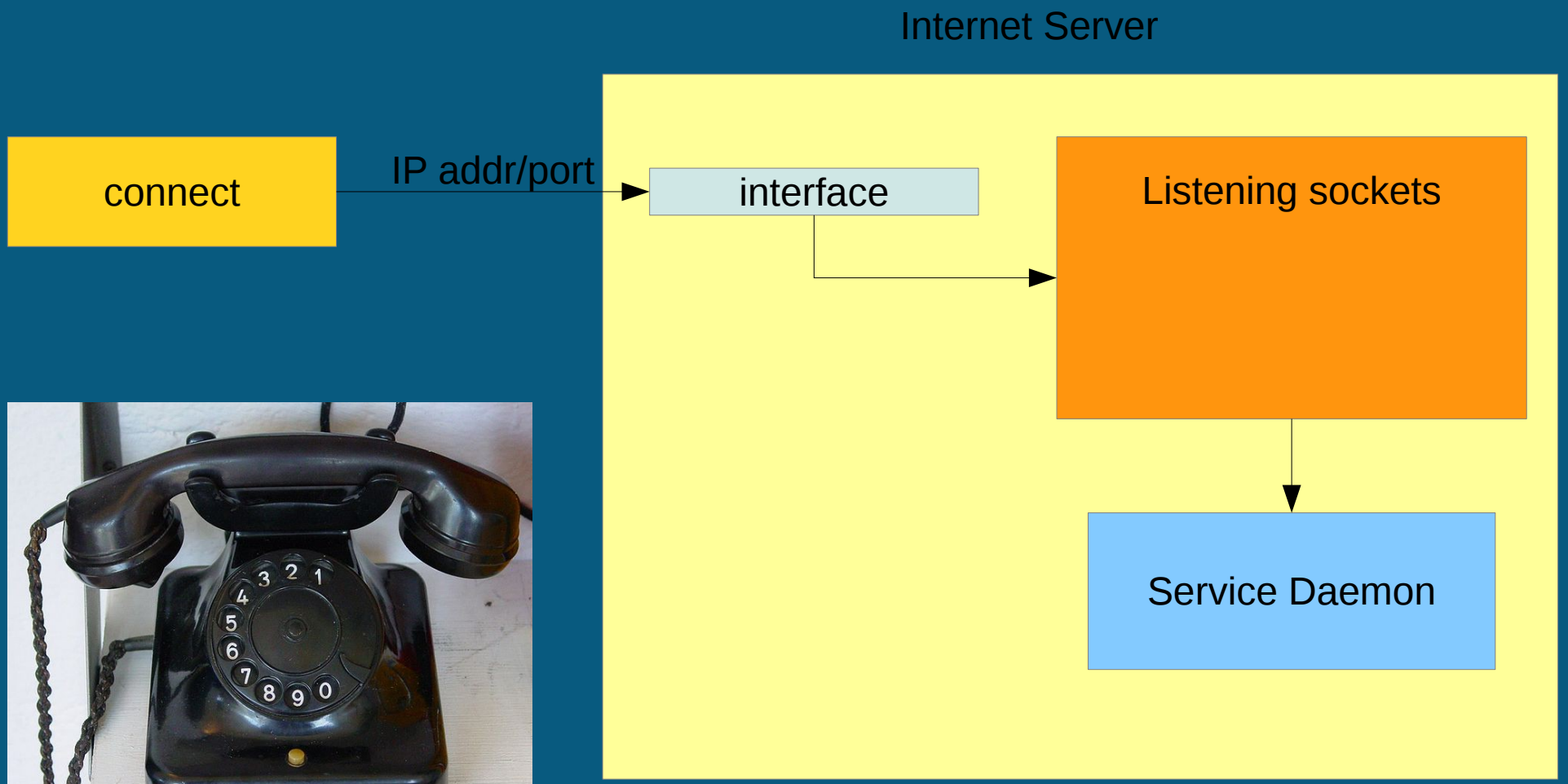
Computer Science Department
University of Bologna

What is an “Internet Node”?

- Network adapters are the “addressable entities” of the Internet.



Yesterday: historical view

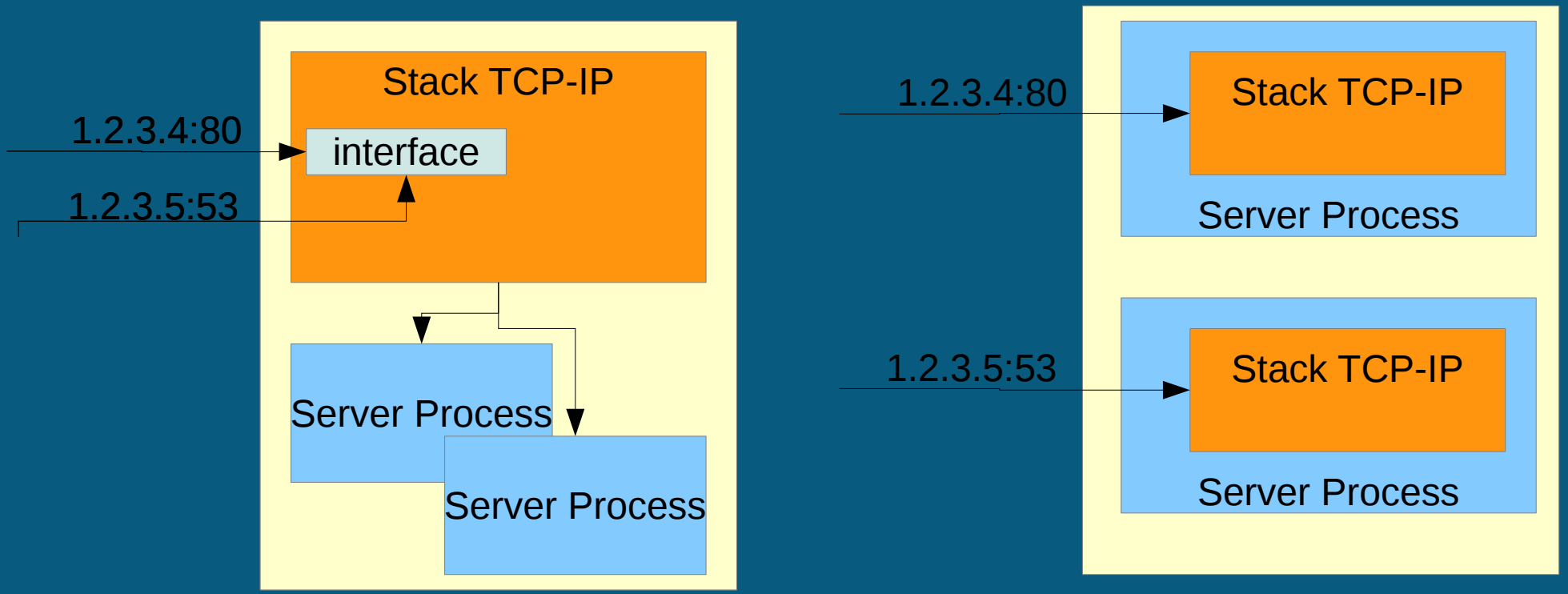


Today: Virtual Internet nodes

- Virtual machines have IP addresses. (xen, kvm, user-mode linux etc)
- High Availability servers have multiple addresses (each service is mapped to a specific address as services must migrate in case of fault. Service addresses are already mapped to interfaces)

Internet of Threads (IoTh)

- Processes, Threads can be Internet nodes (too).
- Each process can use all the networking stacks it needs.



Why the IoTh? Because some “hard” problems in multitasking multiuser systems become simple.

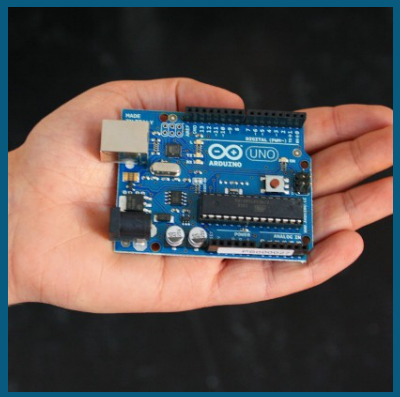
- Provide users working on the same system with user-specific IP addresses
- Provide different QoS/Routing to processes running on the same system
- Run several servers on the same interface+port.
- Use a VPN for some processes and the real network for others.
- Migrate a process to another system while keeping the status of the active connections.

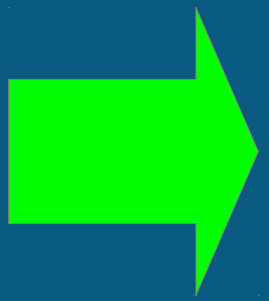
Why the Ioth?

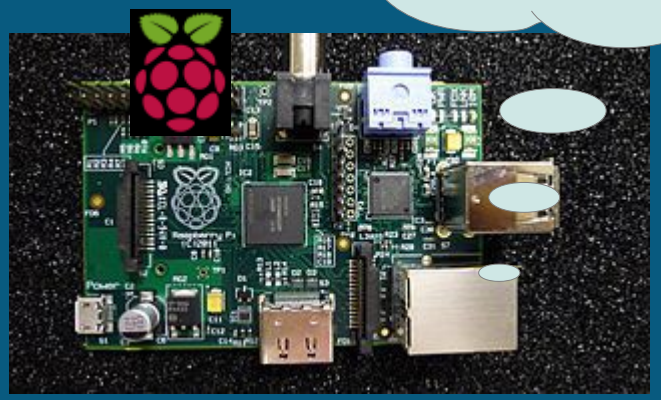
- Because we have IPv6!
- We have plenty of addresses
- We must start again from chalk and blackboards and design a new Internet
- IPv4 network concepts are obsolete
- (we must find an excuse to need IPv8/IPvA ...)

Why the Ioth?

- It is the software counterpart of the Internet of Things.







Services, services, services

- HW vs SW
- Special Purpose vs. General Purpose
- Consumer Electronics vs. Upgradability

Why the IoTh?

- Also Linux Containers (Solaris zones) provide several stacks... but just for sysadm!
- By the IoTh networking becomes “ordinary business” for users.
- Network stacks will be chosen as printers using menu items...

Why the IoTh?

- Because it is natural:
- A network is made of wires, ropes, threads

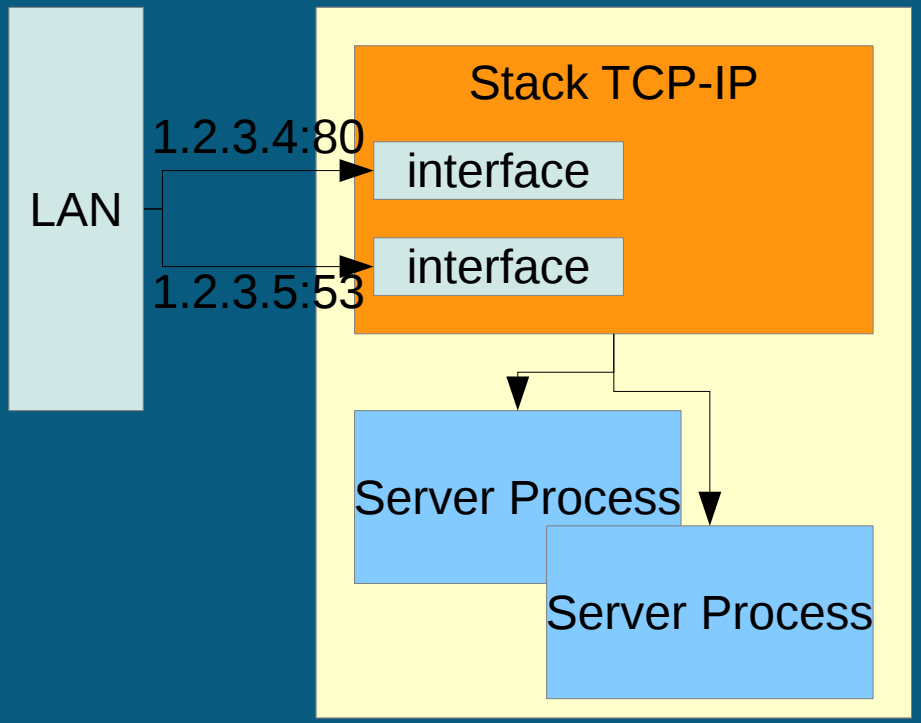
... you cannot knot things together without threads!

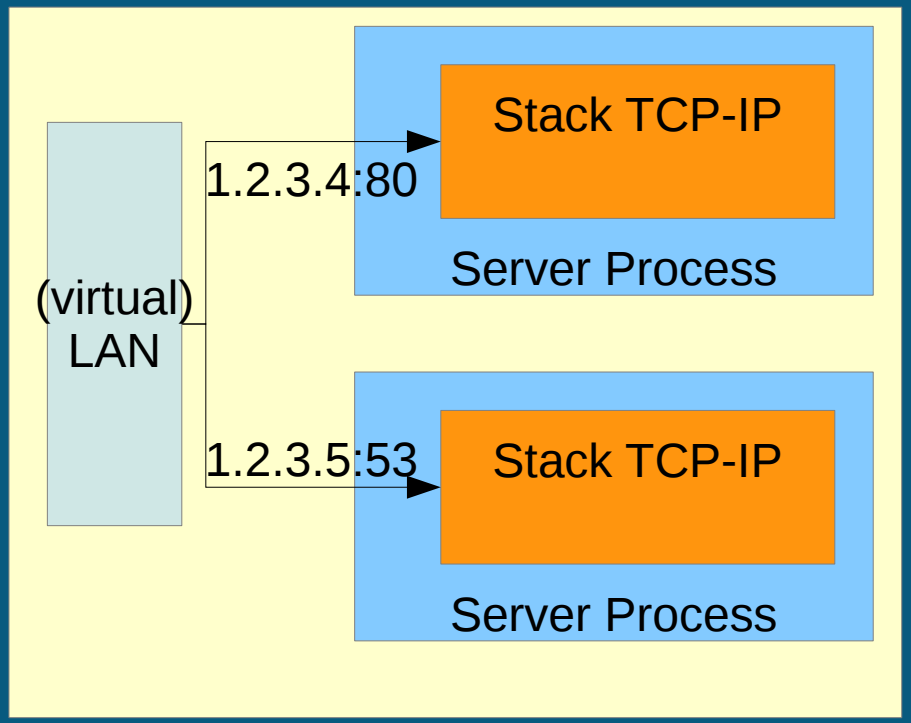
Internet of Threads: examples

- Applications living on different networks
  - e.g. Browsers providing different perspectives on the network
- Virtual appliances/virtual embedded appliances
  - Virtual NAS
  - Web access to user programs
- One Time IP address
  - Repelling the wily port scanner/intruder.

Internet of Threads: where is the LAN?

- The LAN must be virtualized, too





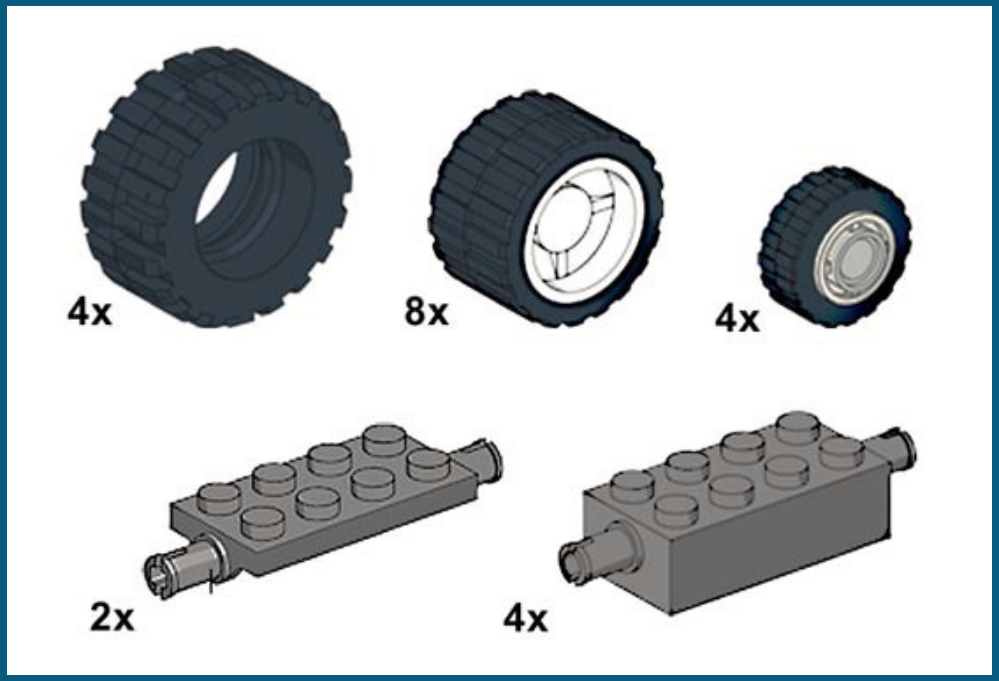
Virtual Ethernet

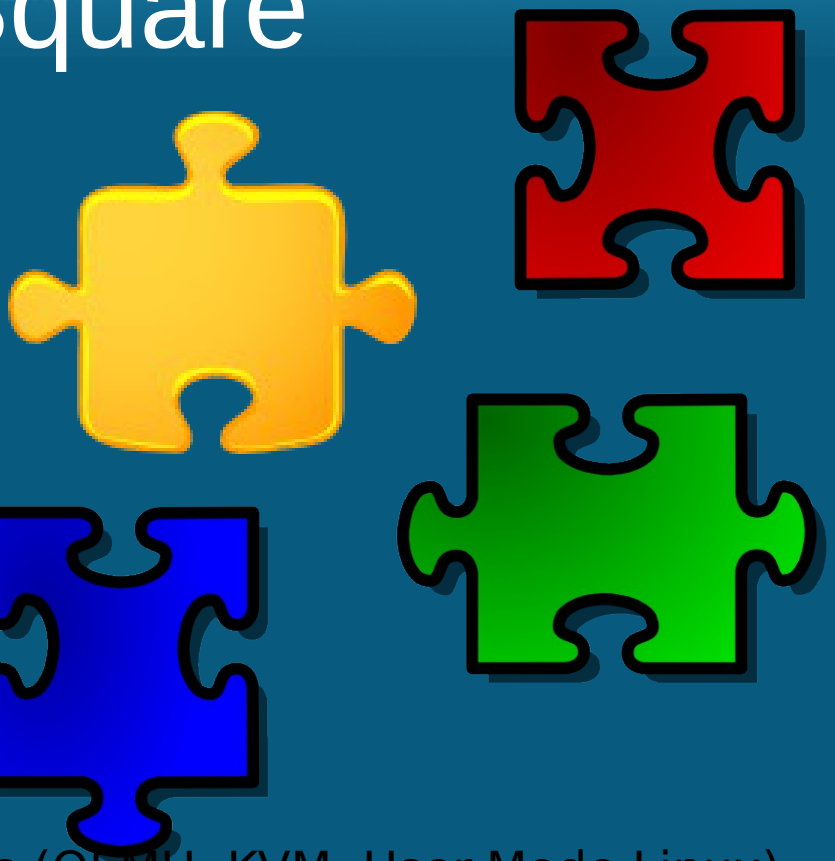
- Processes use virtual networks to communicate and to join the Internet.
- The same tools commonly used in Local Area Networks for hardware nodes (e.g. Personal computers) work on virtual networks to connect processes (e.g. A process can acquire its IP address by DHCP or IPv6 autoconfiguration).
- It is possible to set up hybrid networks, partially real, partially virtual but trasparently connected (virtual and real nodes, machines and processes are undistiguishable from the Internet)

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), ...

FOSDEM

FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

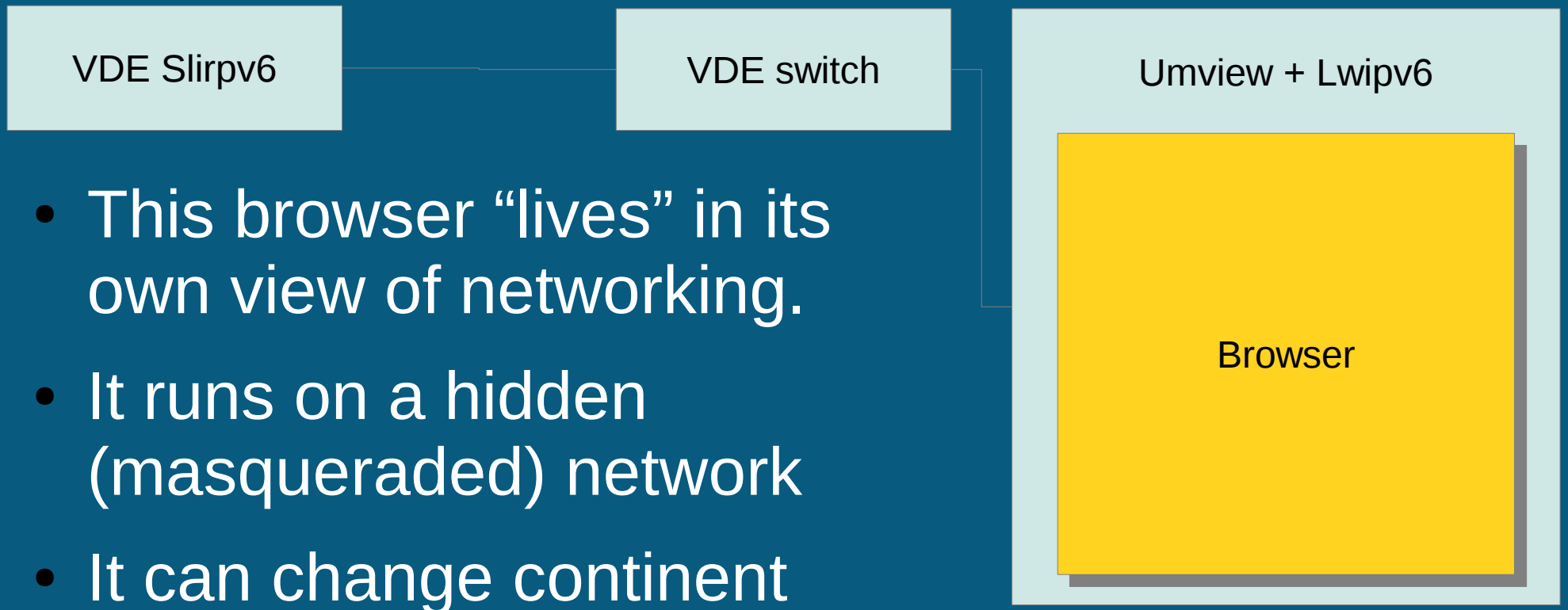
Virtual Square Goals

- 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!

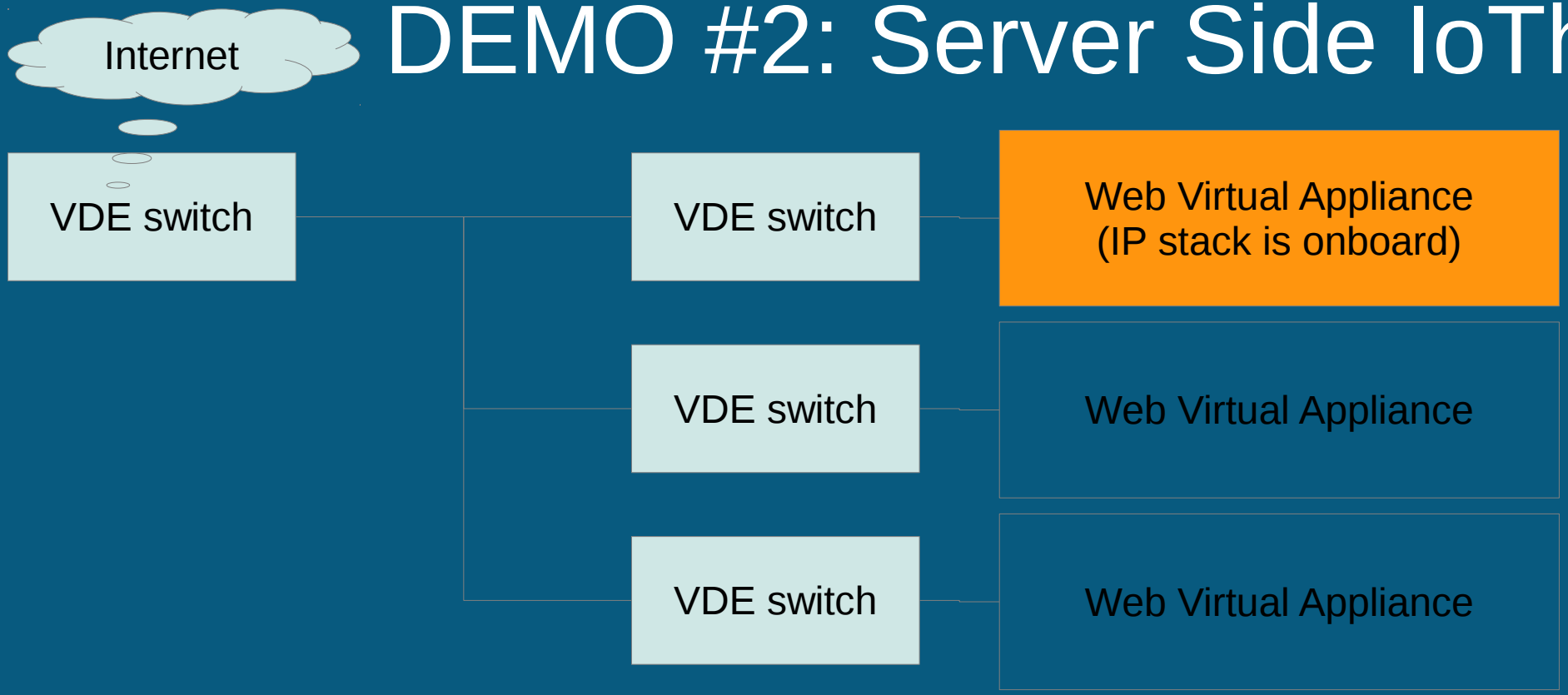
- Virtual Square has developed a set of tools that permit the implementation of the “Internet of Threads” concept:
  - VDE: virtual distributed ethernet
  - LWIPv6: an entire IPv4/v6 hybrid stack as a library
  - View-OS: view-based operating system.
    - Umview/kmview: View-OS implementations as partial virtual machines running on GNU-Linux.
- This is a usable proof-of-concept test set!

DEMO #1: Client side IoTh



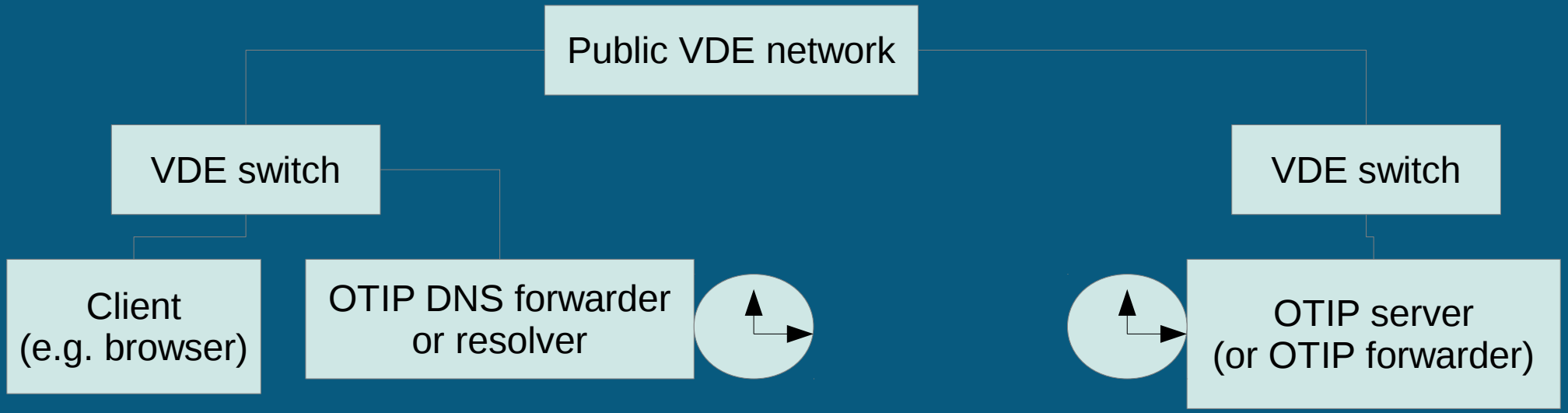
- This browser “lives” in its own view of networking.
- It runs on a hidden (masqueraded) network
- It can change continent just by reloading the page (if we “migrate” the slirp process at the other end)

DEMO #2: Server Side IoTh



- The virtual web appliance has its net stack onboard.
- From the Internet it is (appears as) a web server.
- It can migrate throughout the VDE

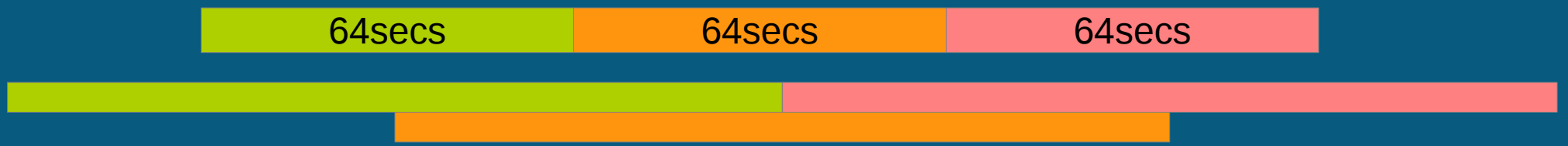
DEMO #3: One Time Internet Protocol (OTIP)



- Server and client changes their address synchronously.
- Repelling the wily port scanner, and the brute force password cracker.

OTIP addresses

- OTIP_address=Prefix::hash(base suffix,password, time)
- The leading 64 bits match the network
- The trailing 64 bits change (almost) synchronously
- The test implementation renews the address each 64 seconds. The servers keep each address for 128 seconds (to tolerate 32 seconds of clock drift and network delays).
- If there are open connections the stack “survives” its deadline to complete these connections. The listening sockets get closed.
- The servers use one stack per address.



One word about security

- Principle of Least Privilege
  - Each process must be given no more privilege than necessary to perform its job.
- Internet of threads:
  - There is no need for “root” access. An overuse of admistrative privileges weaken the infrastructure.
  - The use of specific per service stacks reduces the risks of domino effects caused by weak services.

A word about performance

- Multiple stacks require more memory.
- VDE can dispatch packets at kernel level (already implemented: IPN+kvde_switch).
- The code of the stacks could be shared and can run as a kernel module (already to be designed and implemented)

FOSDEM

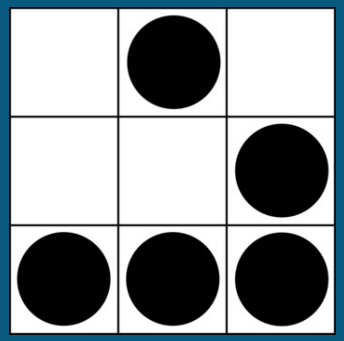
FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

Final remarks

- Network stacks are “ordinary business”. They can be created and configured at user level.
- The Internet of Threads is **the** way to do IPC in the cloud.

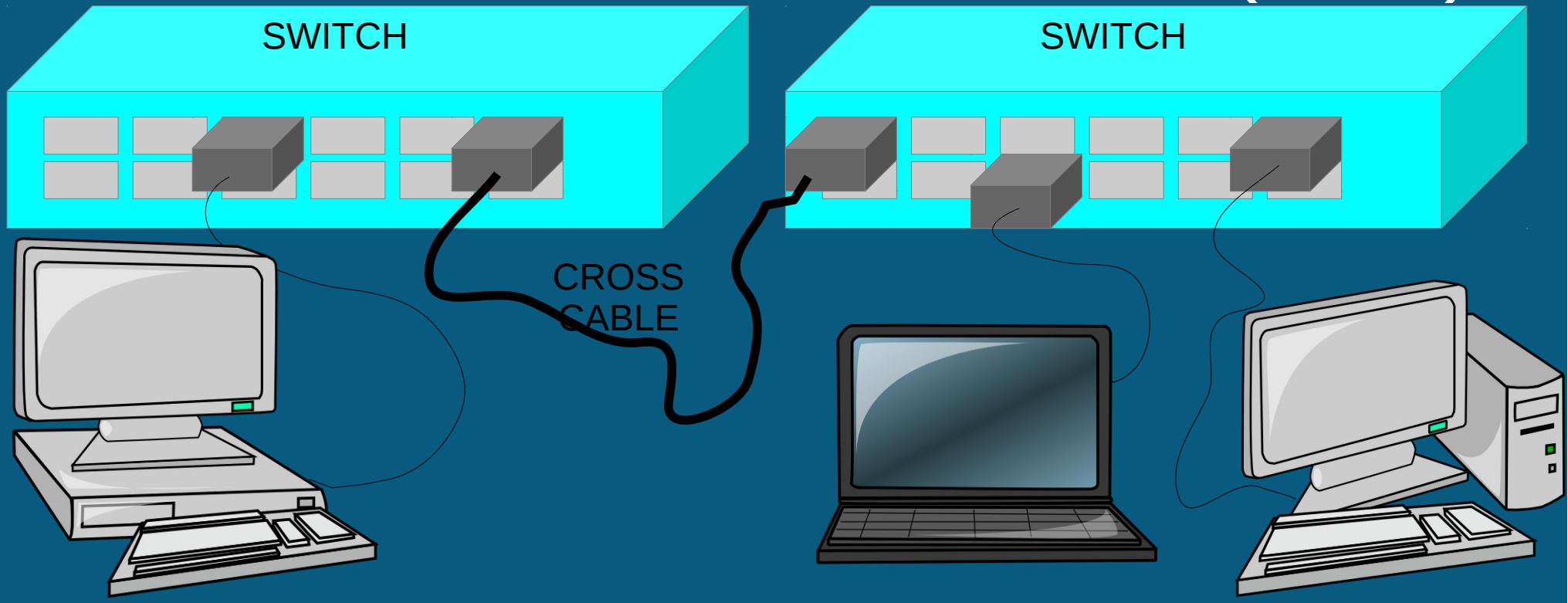
We are still creating art and beauty on a computer:

the art and beauty of revolutionary ideas translated into (libre) code...



renzo, rd235, iz4dje

Virtual Distributed Ethernet (VDE)



VDE SWITCH

VDE SWITCH

VM
(e.g. QEMU)

VDE
plug

VdeWire
(e.g. ssh)

VDE
plug

VM
(e.g. U-ML)

TunTap
Linux Module

VDEv2: advertisement

- VDEv2:
  - modular design
  - compatible with user-mode linux, qemu, kvm, virtualbox, tuntap, (bochs, plex86), umview/lwipv6
  - through the vdetaplib potentially compatible with 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 as defined in socket(2).
- A stack is a special file (new type of special file, see stat(2)):

  #define S_IFSTACK 0160000
- Each process has a default stack for each protocol family (domain).
  - If **path==NULL**, msocket uses the default stack.
- It is backwards compatible.

  #define socket(d,t,p) msocket(NULL,(d),(t),(p))

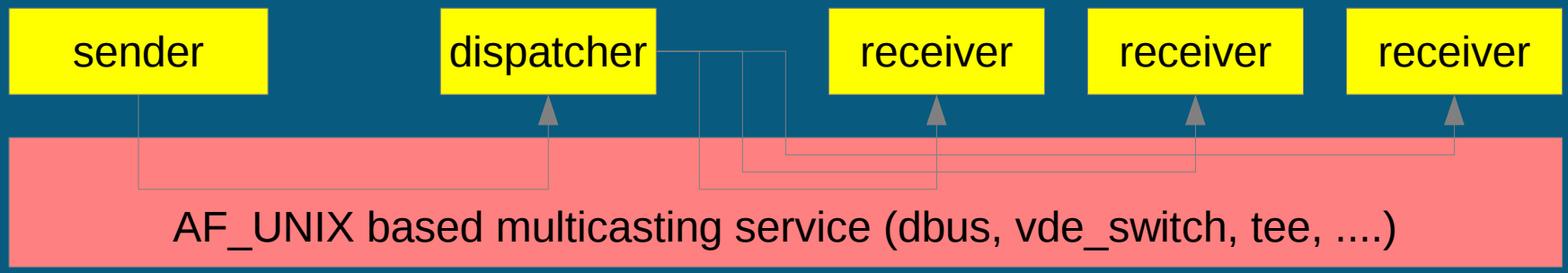
Berkeley sockets API: problem #2

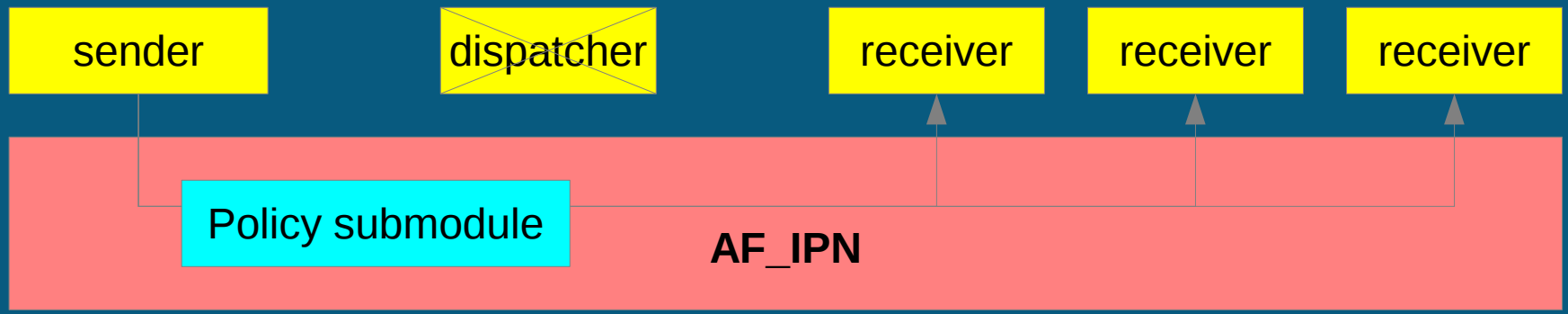
- Berkeley Sockets API does provide support for IPC (AF_UNIX)
- Berkeley Sockets API does not provide support for multicast IPC
- Berkeley Sockets is mainly for point-to-point, client-server communication

  IP multicast, Ethernet broadcast provided by “magic” addresses.
- 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.

FOSDEM

FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

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
  - ...

FOSDEM

FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

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....)

FOSDEM
FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

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.

FOSDEM

FREE AND OPEN SOURCE SOFTWARE DEVELOPERS' EUROPEAN MEETING

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
$
```