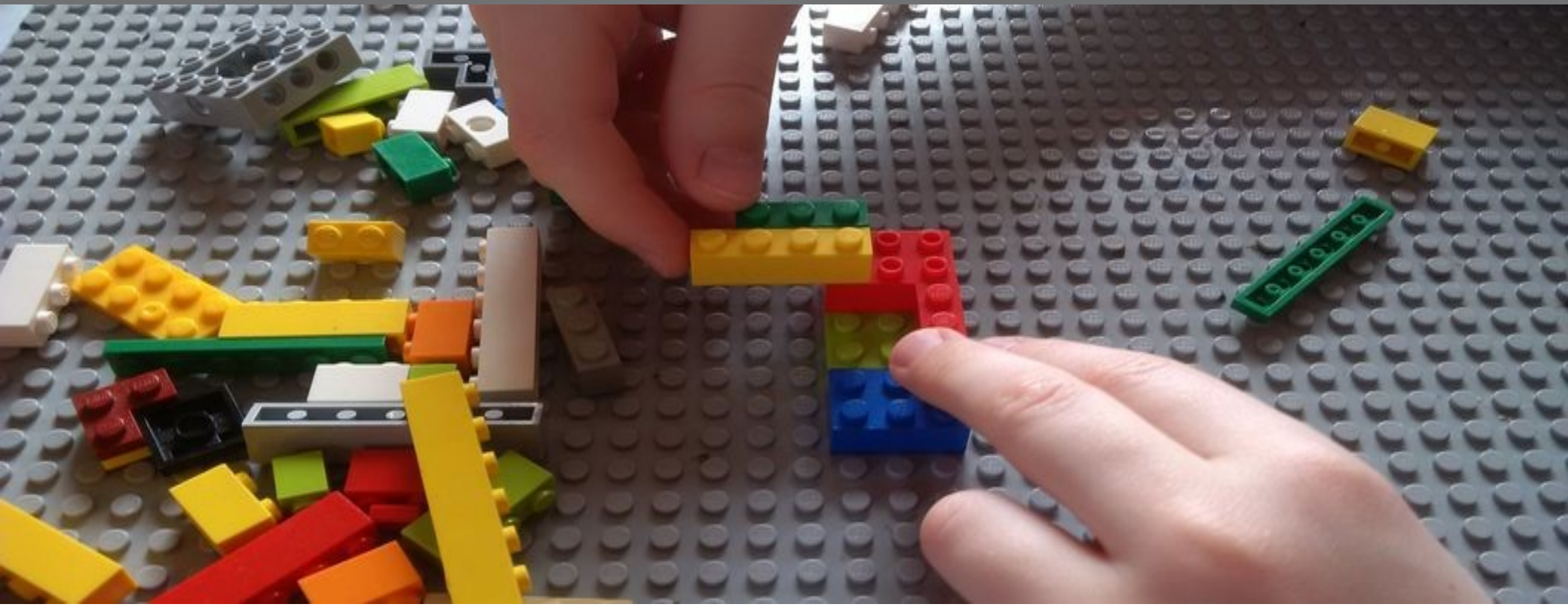


Hands-on composition of basic L4Re components

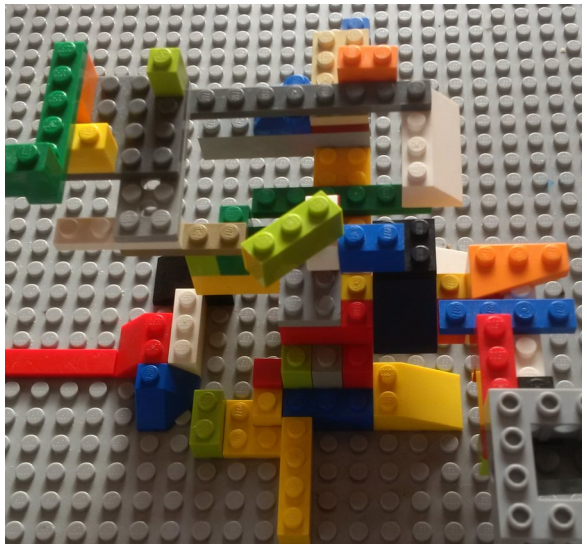
Jakub Jermář
Kernkonzept GmbH

L4Re

L4 Runtime Environment

Not a microkernel distribution

Set of components from which everything can be built



So what can it do?

Sky is the limit

Very often used in virtualization scenarios

Let’s start with something simple

Assume all prerequisities are built

Will use QEMU 64-bit ARM virtual platform

Hello world!

Module list

```
entry hello-ned
roottask moe rom/hello.ned
module l4re
module ned
module hello.ned
module hello
```

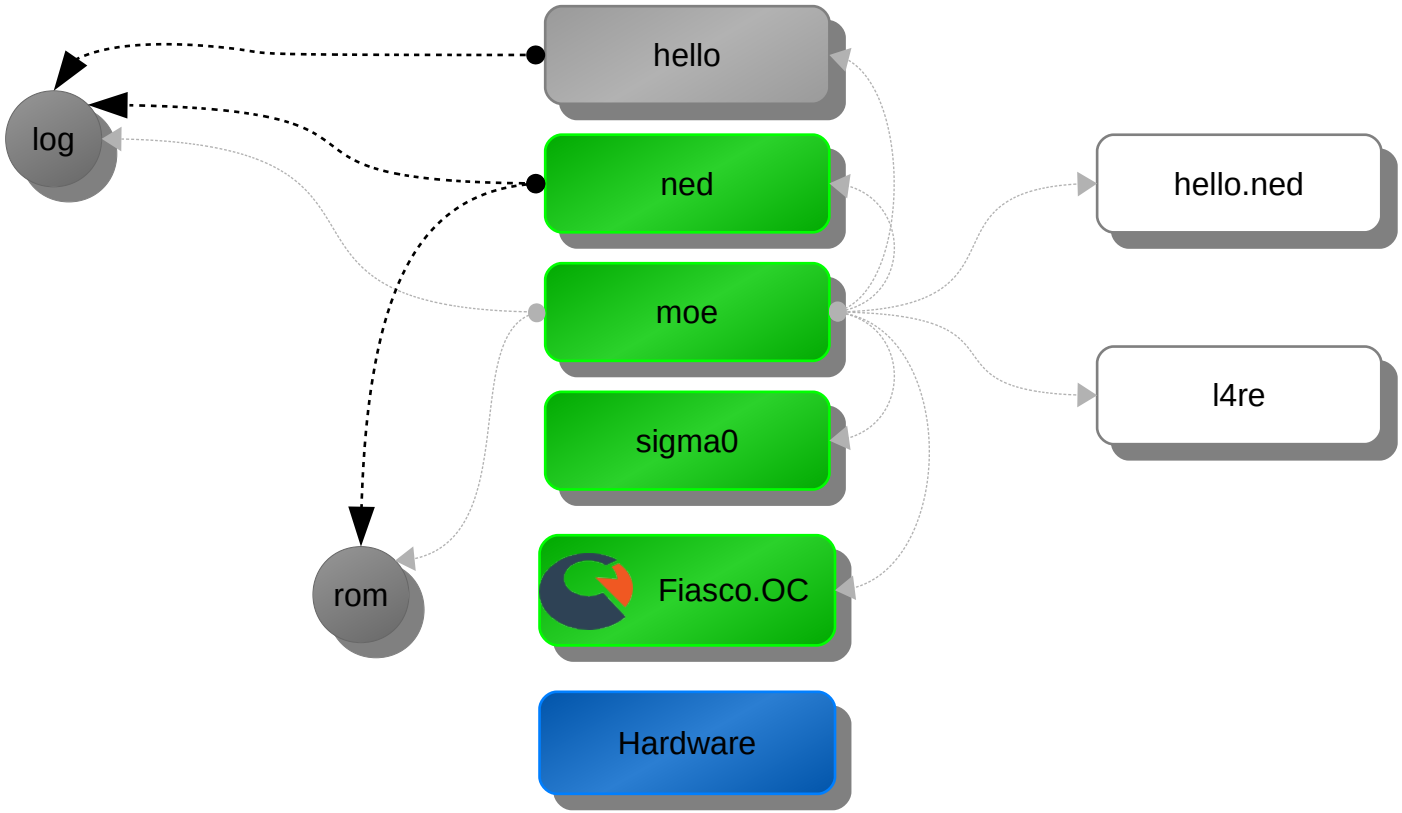
Ned scripts

Embedded Lua interpreter

hello.ned:

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

Root and init tasks



Demo

Running a Linux VM

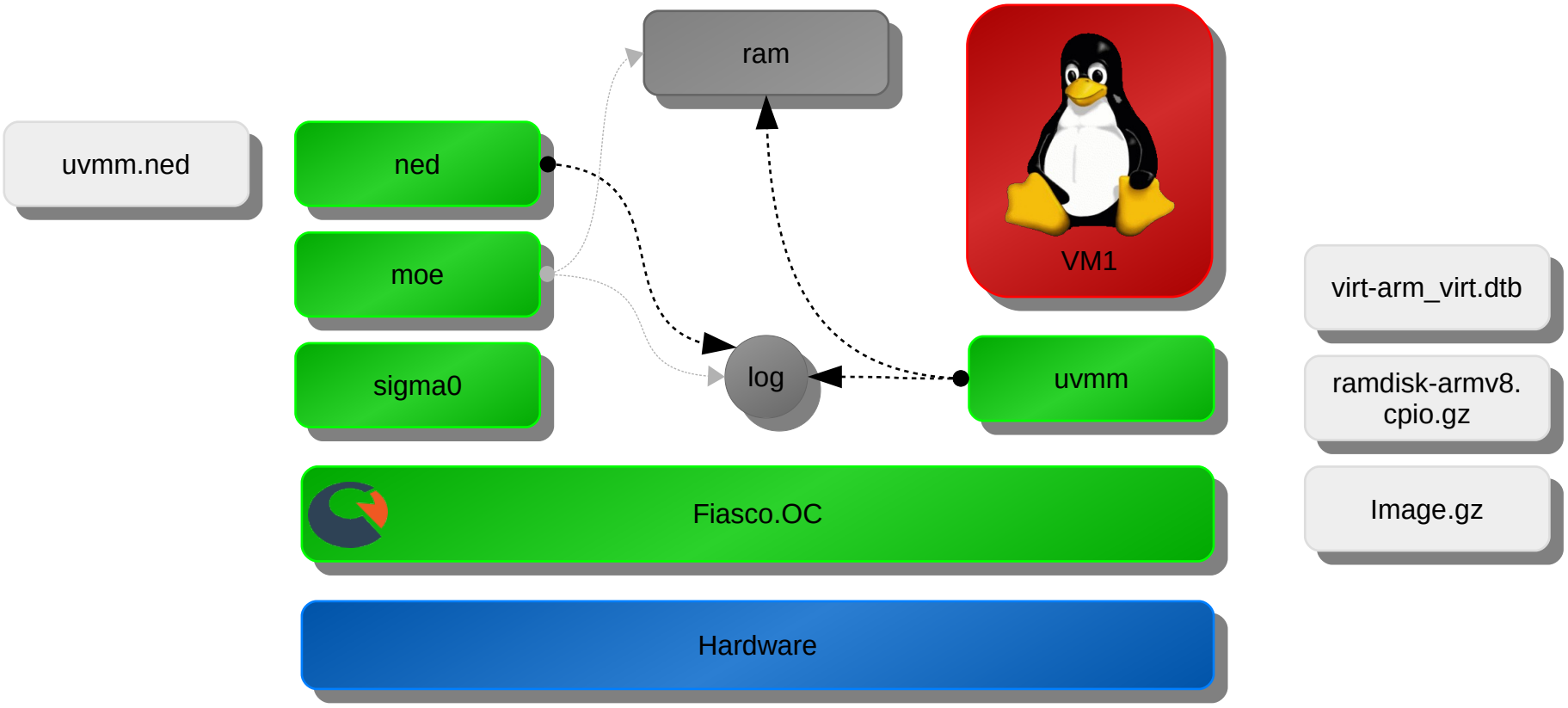
Module list

```
entry uvmm
roottask moe rom/uvmm.ned
module l4re
module ned
module uvmm.ned
module uvmm
module virt-arm_virt.dtb
module ramdisk-armv8.cpio.gz
module Image.gz
```

Ned script

```
local L4 = require "L4";
L4.default_loader:startv({
  log = L4.Env.log,
  caps = {
    ram  = L4.Env.user_factory:create(
      L4.Proto.Dataspace,
      128 * 1024 * 1024,                                  -- size in MB
      L4.Mem_alloc_flags.Continuous |
        L4.Mem_alloc_flags.Pinned |
        L4.Mem_alloc_flags.Super_pages,
      21                                                  -- alignment
    ):m("rws");
  }
}, "rom/uvmm", "--dtb", "rom/virt-arm_virt.dtb",
  "--ramdisk", "rom/ramdisk-armv8.cpio.gz",
  "--kernel", "rom/Image.gz",
  "--cmdline", "console=hvc0 earlyprintk=1 rdinit=/init");
```

Running a Linux VM



Demo

Running two Linux VMs

Module list

```
entry uvmm-2vm
roottask moe rom/uvmm-2vm.ned
module l4re
module uvmm
module ned
module uvmm-2vm.ned
module uvmm
module cons
module[shell] echo $SRC_BASE_ABS/pkg/uvmm/configs/vmm.lua
module virt-arm_virt.dtb
module ramdisk-armv8.cpio.gz
module Image.gz
```

Running two Linux VMs

VMM package to abstract away complexity

```
package.path = "rom/?.lua";

local L4 = require "L4";
local vmm = require "vmm";

vmm.loader.log_fab = L4.default_loader:new_channel();

local function vm(id, net, args, vbus)
  vmm.start_vm({
    id = id, mem = 128, mon = false, net = net, vbus = vbus,
    rd = "rom/ramdisk-armv8.cpio.gz",
    fdt = "rom/virt-arm_virt.dtb", kernel = "rom/Image.gz",
    bootargs = "console=hvc0 earlyprintk=1 rdinit=/init " .. (args or ""),
  });
end
```

Console multiplexer

Factory pattern

Create a console factory

```
L4.default_loader:start({
    log = L4.Env.log,
    caps = {
      cons = vmm.loader.log_fab:svr();
    }
  }, "rom/cons -a");
```

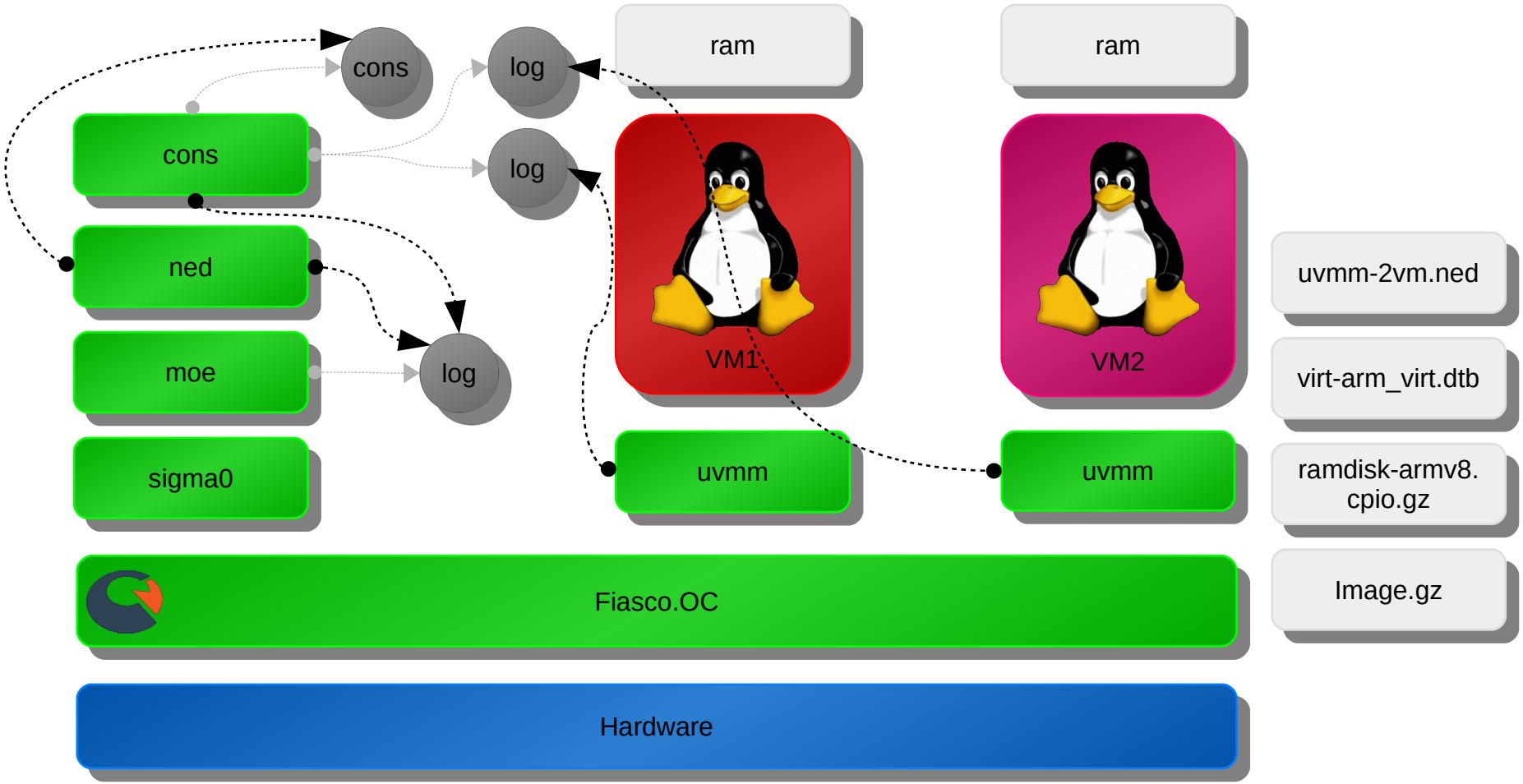
vmm.lua uses the factory to create log for each VM

```
    log = l.log_fab:create(L4.Proto.Log, "vm" .. nr, "w", keyb_shortcut);
    ...
  }, "rom/uvmm", ...
```

Running two Linux VMs

```
...
vm(1);
vm(2);
```

Running two Linux VMs



Demo

Internetworking 2 VMs

Module list

```
entry uvmm-2vm-net
roottask moe rom/uvmm-2vm-net.ned
module l4re
module uvmm
module ned
module uvmm-2vm-net.ned
module uvmm
module cons
module l4vio_net_p2p
module[shell] echo $SRC_BASE_ABS/pkg/uvmm/configs/vmm.lua
module virt-arm_virt.dtb
module ramdisk-armv8.cpio.gz
module Image.gz
```

VIRTIO switch

Factory pattern used in vmm.lua

```
function start_virtio_switch(ports, ...)
  local caps = {};
  local switch = l:new_channel();

  local opts = {
    log = { "switch", "Blue" },
    caps = { svr = switch:svr() };
  };

  svr = l:start(opts, "rom/l4vio_net_p2p");

  for k, v in pairs(ports) do
    ports[k] = L4.cast(L4.Proto.Factory, switch):create(0, 4);
  end
end
```

VIRTIO switch

VIRTIO-net port is passed to each VM

```
function start_vm(options)
  local vbus    = options.vbus;
  local vnet    = options.net;
  ...
  local caps = {
    vbus = vbus;     -- remember this for direct HW pass-through
    net  = vnet;
    ...
  };
  ...
  return l:startv({..., caps}, "rom/uvmm", ...);
end
```

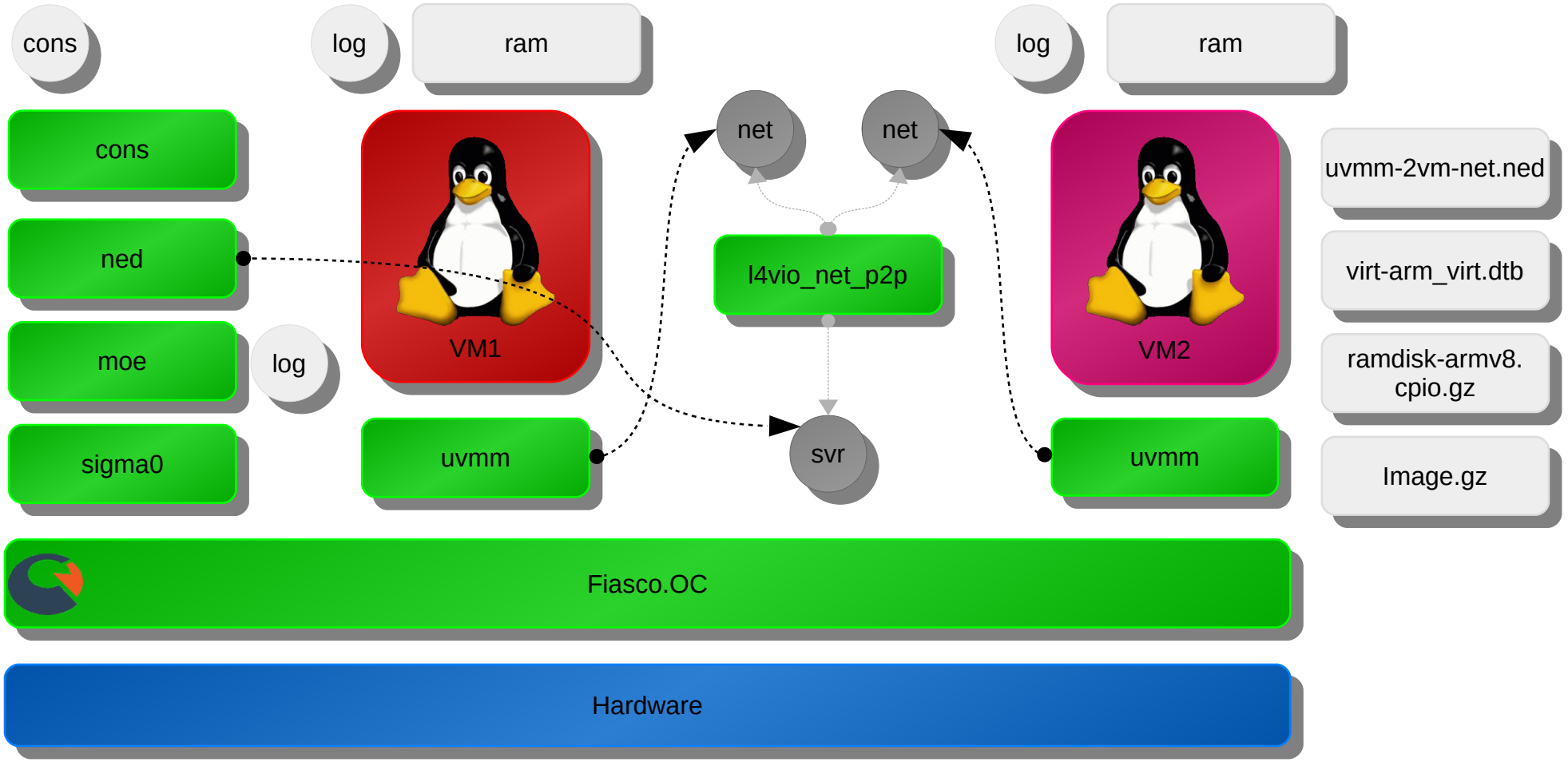
Ned script

```
...

local net_ports = {
  net0 = 1,
  net1 = 1,
}

vmm.start_virtio_switch(net_ports);

vm(1, net_ports.net0, "ip=192.168.1.1:::255.255.255.0:server:eth0");
vm(2, net_ports.net1, "ip=192.168.1.2:::255.255.255.0:server:eth0");
```

Extra device tree nodes

```
/ {
    l4vmm {
        ...
        virtio_net@10000 {
            compatible = "virtio,mmio";
            reg = <0x10000 0x200>;
            interrupt-parent = <&gic>;
            interrupts = <0 123 4>;
            l4vmm,vdev = "proxy";
            l4vmm,virtiocap = "net";  /* virtio switch port cap name */
            l4vmm,no-notify = <1>;
        };
    };
};
```

Internetworking 2 VMs



Demo

HW pass-through

Module list

```
entry uvmm-2vm-net-io
roottask moe rom/uvmm-2vm-net-io.ned
module l4re
module uvmm
module ned
module uvmm-2vm-net-io.ned
module uvmm
module cons
module l4vio_net_p2p
module io
module io.cfg
module vm_hw.vbus
module[shell] echo $SRC_BASE_ABS/pkg/uvmm/configs/vmm.lua
module virt-arm_virt.dtb
module ramdisk-armv8.cpio.gz
module Image.gz
```

Device tree nodes

```
/ {
    apb-pclk {
        phandle = <0x8000>;
        clock-output-names = "clk24mhz";
        clock-frequency = < 0x16e3600 >;
        #clock-cells = < 0x00 >;
        compatible = "fixed-clock";
    };
    pl031@9010000 {
        clock-names = "apb_pclk";
        clocks = <0x8000>;
        interrupts = < 0x00 0x02 0x04 >;
        reg = < 0x00 0x9010000 0x00 0x1000 >;
        compatible = "arm,pl031", "arm,primecell";
    };
};
```

Ned script

```
local net_ports = {
  net0 = 1,
  net1 = 1,
}

local io_busses = {
  vm_hw = 1,
}

vmm.start_io(io_busses, "rom/io.cfg");
vmm.start_virtio_switch(net_ports);

vm(1, net_ports.net0, "ip=192.168.1.1:::255.255.255.0:server:eth0",
  io_busses.vm_hw);
vm(2, net_ports.net1, "ip=192.168.1.2:::255.255.255.0:server:eth0");
```

Virtual busses

```
function start_io(busses, opts)
  local caps = {
    sigma0 = L4.cast(L4.Proto.Factory, L4.Env.sigma0):create(L4.Proto.Sigma0);
    icu    = L4.Env.icu;
  };
  local files = "";
  for k, v in pairs(busses) do
    local c = l:new_channel();
    busses[k] = c                    -- passed to client (uvmm) as "vbus"
    caps[k] = c:svr();               -- passed to io as k (e.g. "vm_hw")
    files = files .. " rom/" .. k .. ".vbus";
  end
  return l:start({
    caps = caps
  }, "rom/io " .. opts .. files)
end
```

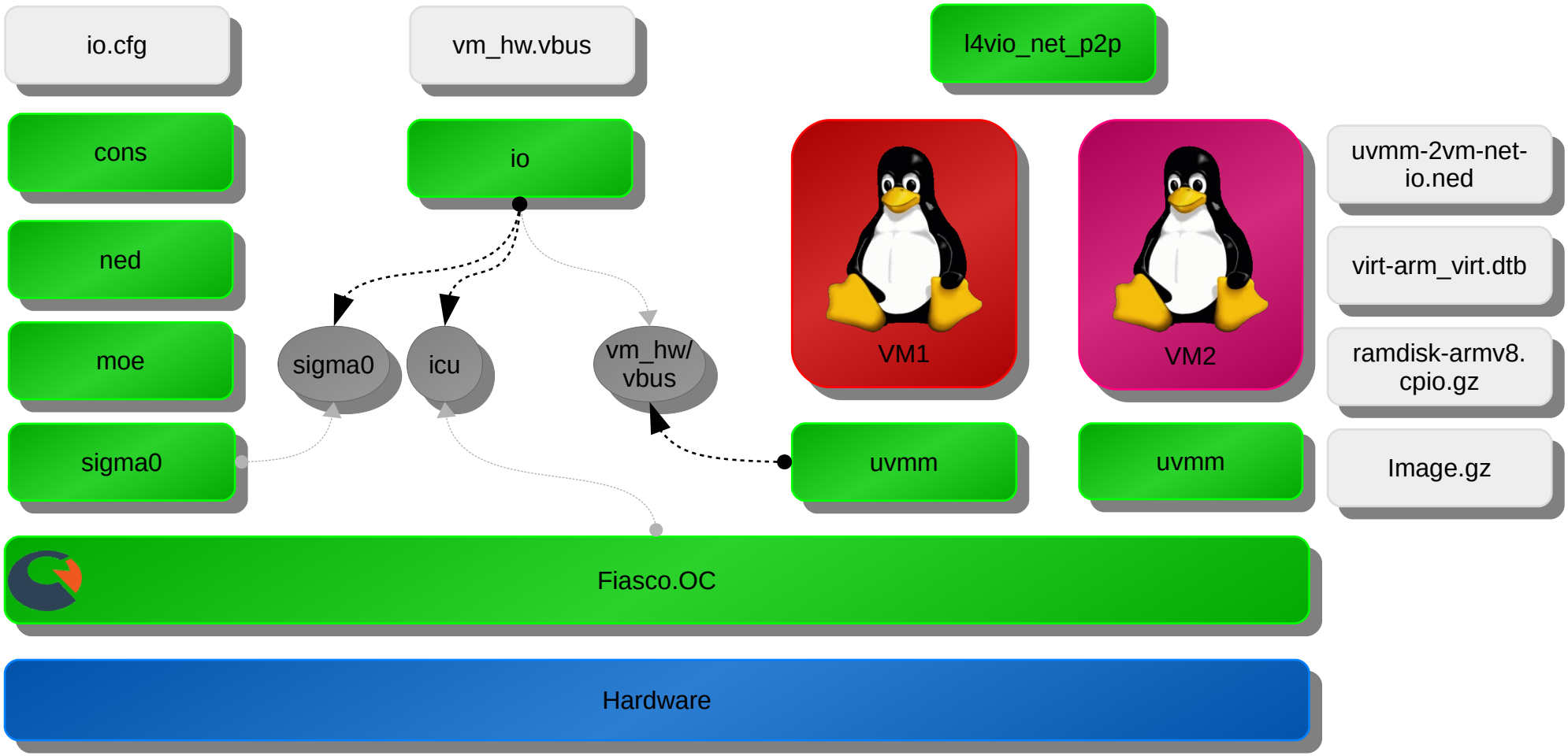
IO config

```
local Res = Io.Res
local Hw = Io.Hw

local add_children = Io.Dt.add_children

add_children(Io.system_bus(), function()
  rtc = Hw.Device(function()
    compatible = { "arm,pl031" };
    Resource.reg0 = Res.mmio(0x9010000, 0x9010fff);
    Resource.irq0 = Res.irq(32 + 2, Io.Resource.Irq_type_level_high);
  end)
end)
```

VBUS config

```
Io.add_vbusses
{
  vm_hw = Io.Vi.System_bus(function()
    PL031 = wrap(Io.system_bus().rtc);
  end);
}
```

HW pass-through



Demo

Where to get L4Re?

GitHub

https://github.com/kernkonzept

Snapshot

http://l4re.org/download.html

Tutorials

https://github.com/kernkonzept/manifest/wiki#tutorials

Takeaway

C.I.A.

Components

Interconnected via capabilities

And configured in Lua

(quite heavy use of VIRTIO)

Thank you