

Cursul 10

Virtualizare



Objective

- Conceptul de virtualizare
- Virtualizarea rețelei în medii cloud
- Virtualizarea rețelei în Linux
- Rețele cu VMware
- Rețele cu LXC
- SDN

Introducere în virtualizare

- Termeni
- Clasificarea soluțiilor de virtualizare
- Avantajele virtualizării



Termeni

Virtualizare

- Mecanism prin care se creează o entitate cu (aproape) toate funcționalitățile unei entități fizice, fără ca aceasta să existe fizic

Hypervisor

- Entitatea care implementează și controlează virtualizarea
- Permite crearea / distrugerea / gestiunea resurselor virtualizate

Termeni

Mașină fizică

- Termen folosit pentru platforma hardware pe care rulează mecanismul de virtualizare

Mașină virtuală

- Termen folosit pentru denumirea entităților virtuale ce reprezintă sisteme întregi (de obicei folosit pentru stații virtuale)

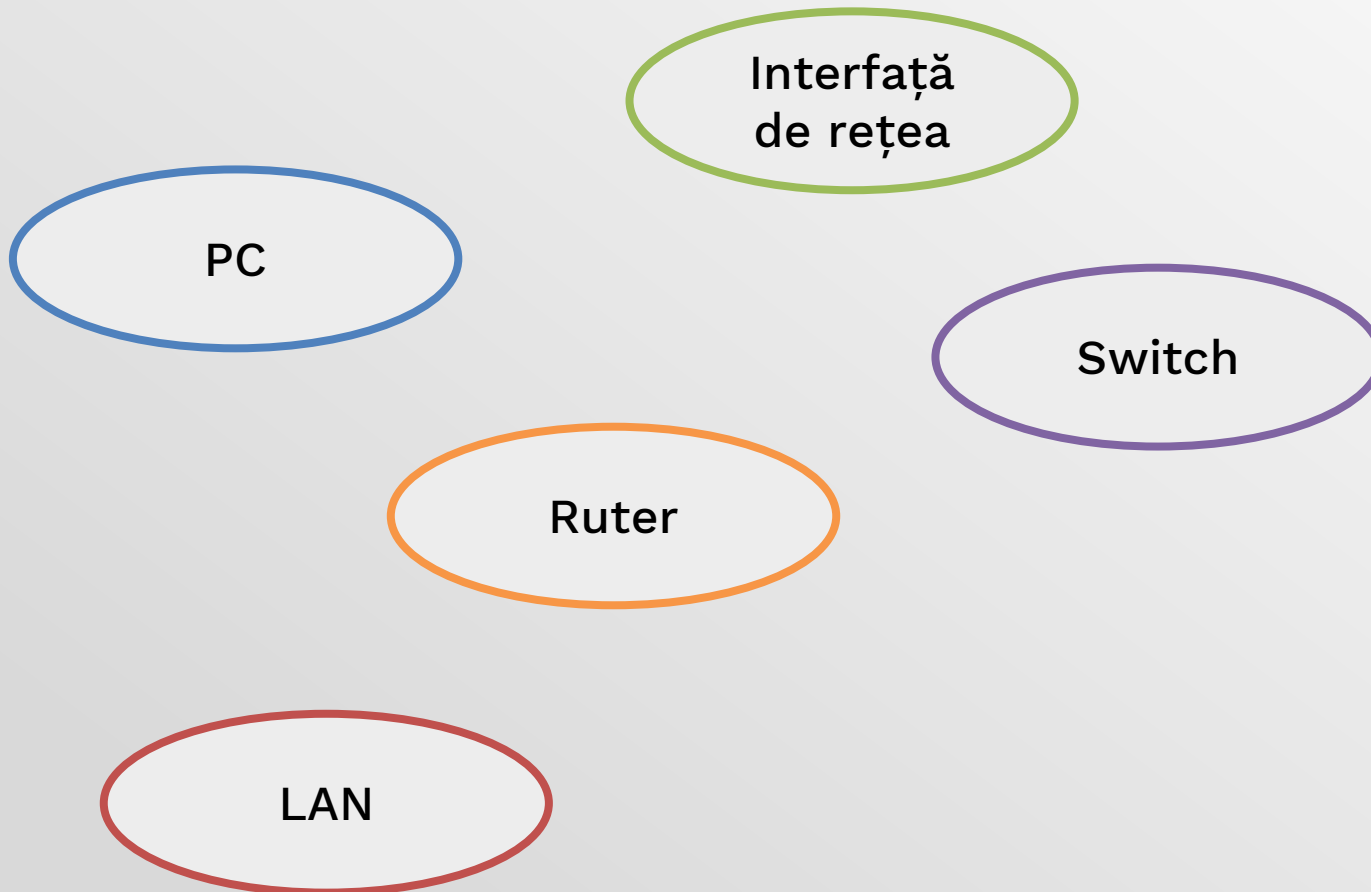
NIC

- Network Interface Card; o placă de rețea fizică

vNIC

- Virtual Network Interface Card; o placă de rețea virtualizată

Ce se poate virtualiza



Avantajele virtualizării

- Consolidarea infrastructurii
 - Pe aceeași rețea fizică coexistă multiple rețele virtuale ce utilizează eficient resursele disponibile
- Flexibilitate
 - Componentele virtuale pot fi create sau distruse cu ușurință, în funcție de obiectivele existente
 - Gestionarea componentelor virtuale este simplă, comparativ cu gestionarea unor componente fizice
- Securitate
 - Comunicarea poate fi izolată sau criptată



Virtualizarea în cloud

- Provocări
- VXLAN
- NVGRE



RL
crunch it connected

Provocările virtualizării în cloud

Transparență

Clienții cloud-ului nu trebuie să fie preocupați de complexitățile virtualizării.

Eficiență

Resursele fizice ale cloud-ului trebuie să fie exploatate eficient de către sistemele de virtualizare.

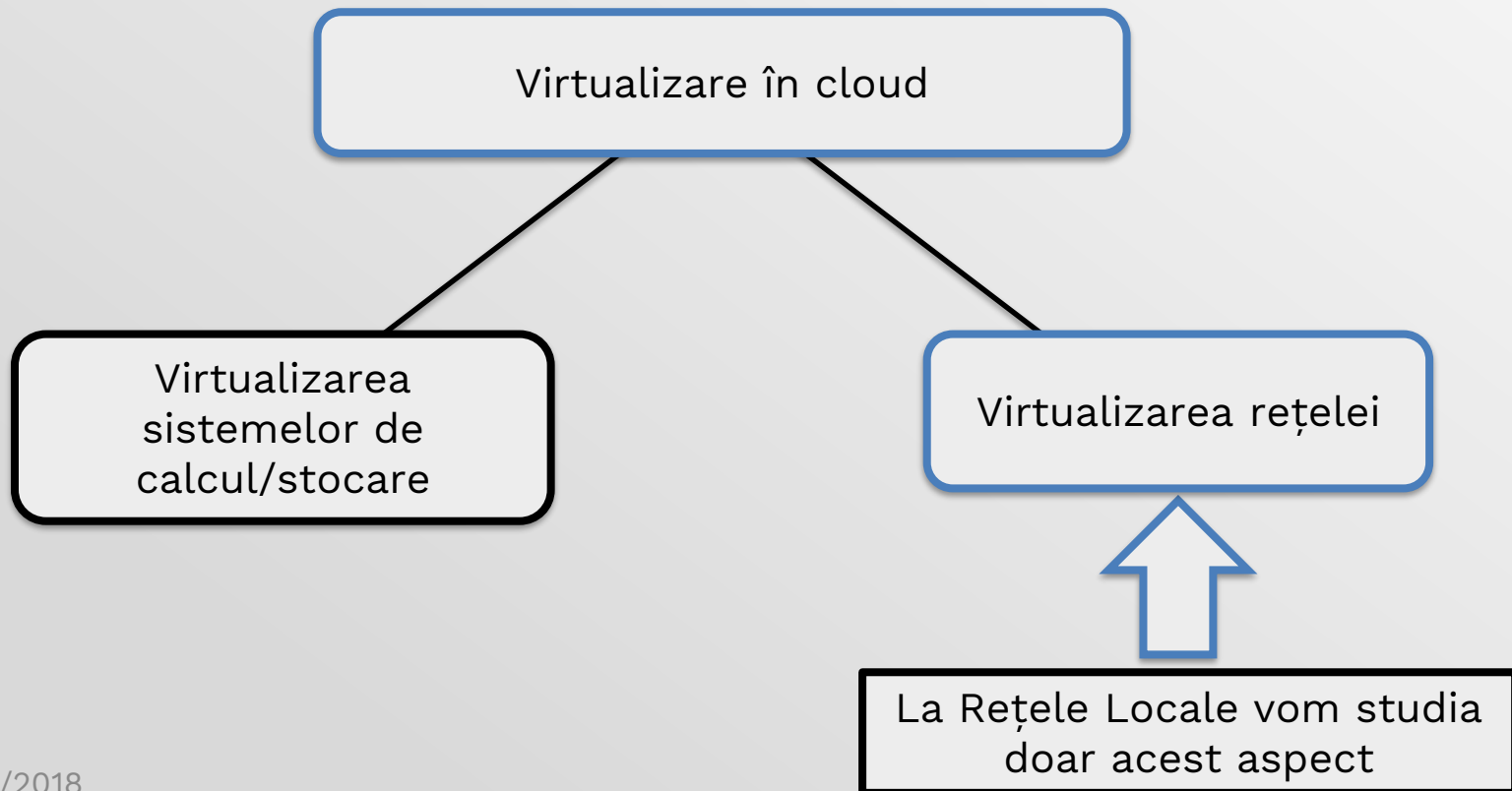
Izolare

Un client nu trebuie să poată afecta operațiunile altui client.

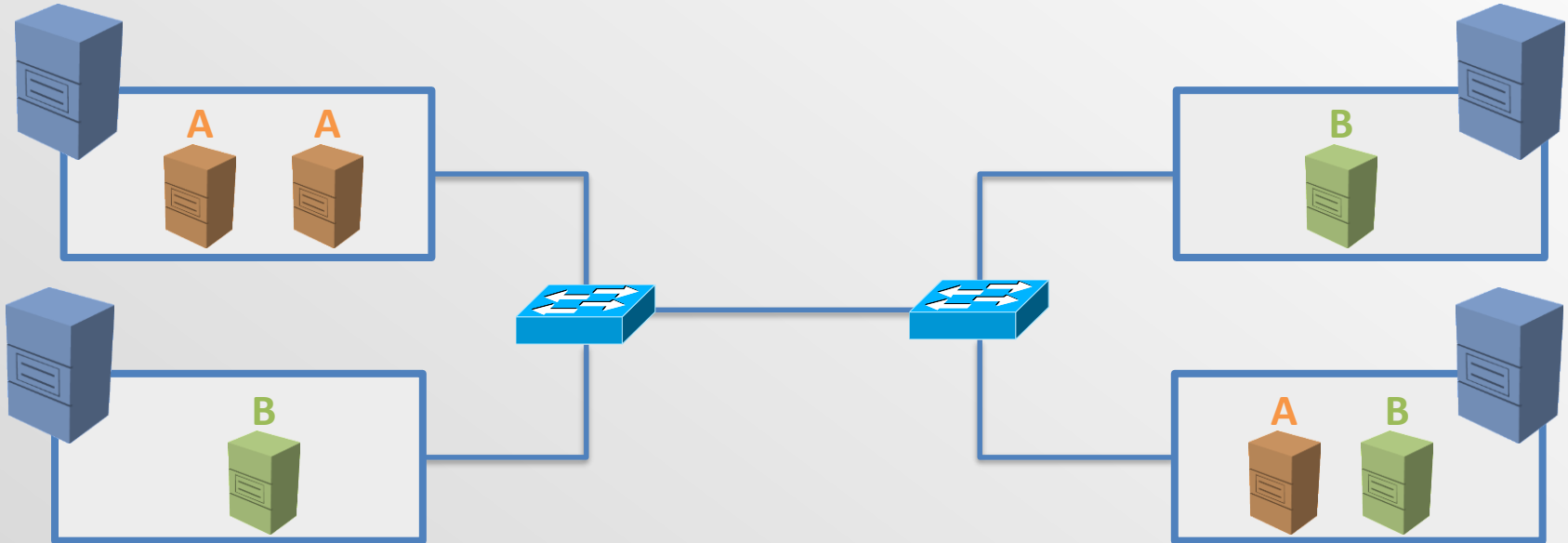
Aceste provocări au dus la dezvoltarea puternică din ultimii ani a tehnologiilor de virtualizare

Provocările virtualizării în cloud

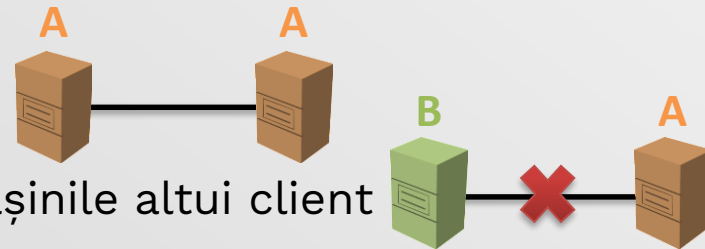
- Virtualizarea în cloud poate fi structurată în două componente:



Virtualizarea rețelei în cloud



- Mașinile unui client trebuie:
 - Să aibă conectivitate între ele
 - Să nu aibă conectivitate cu mașinile altui client



- Soluție: **VLAN-uri**

- Dar dacă avem mai mulți clienți decât există VLAN-uri?

Soluția 1: VXLAN

- Actual draft IETF
- Similar în funcționalitate cu VLAN
 - Partiționează o rețea în mai multe subrețele distincte private
- VLAN folosește pentru identificare câmpul VLANID pe 12 biți
 - permite cel mult $2^{12} = 4096$ de rețele diferite
- VXLAN folosește pentru identificare câmpul **VNI** (Virtual Network Identifier) pe 24 de biți
 - permite cel mult $2^{24} = 16$ milioane de rețele diferite
- VXLAN încapsulează pachetele mașinilor virtuale în segmente UDP
- Hypervisor-ul trebuie să fie capabil să prelucreze antete VXLAN

VXLAN

- **VM1** și **VM3** aparțin aceluiași client cloud (cu **VNI = 42**)
- Fiecare hypervisor reține această informație în tabela VXLAN

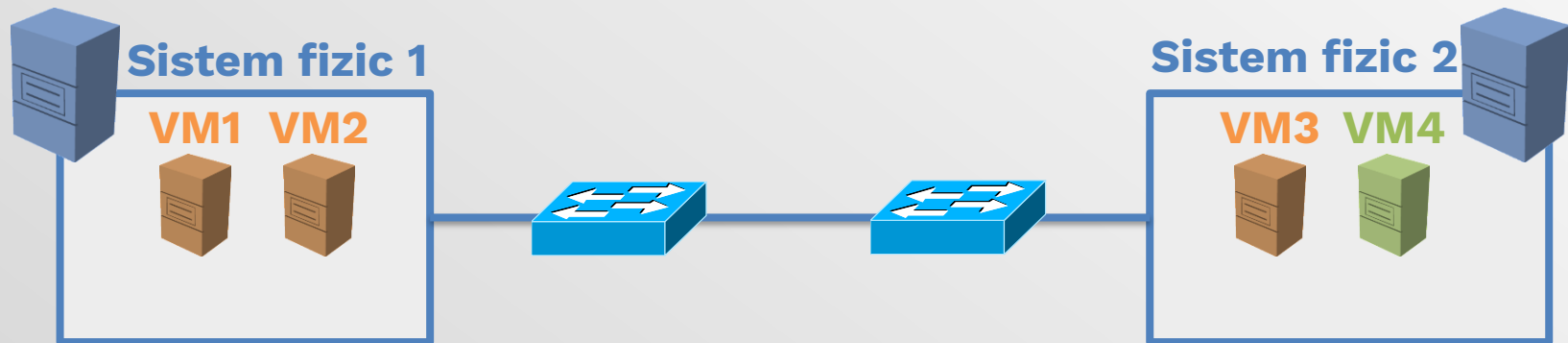


Tabela VXLAN

VNI = 42: VM1, VM2

VNI = 73:

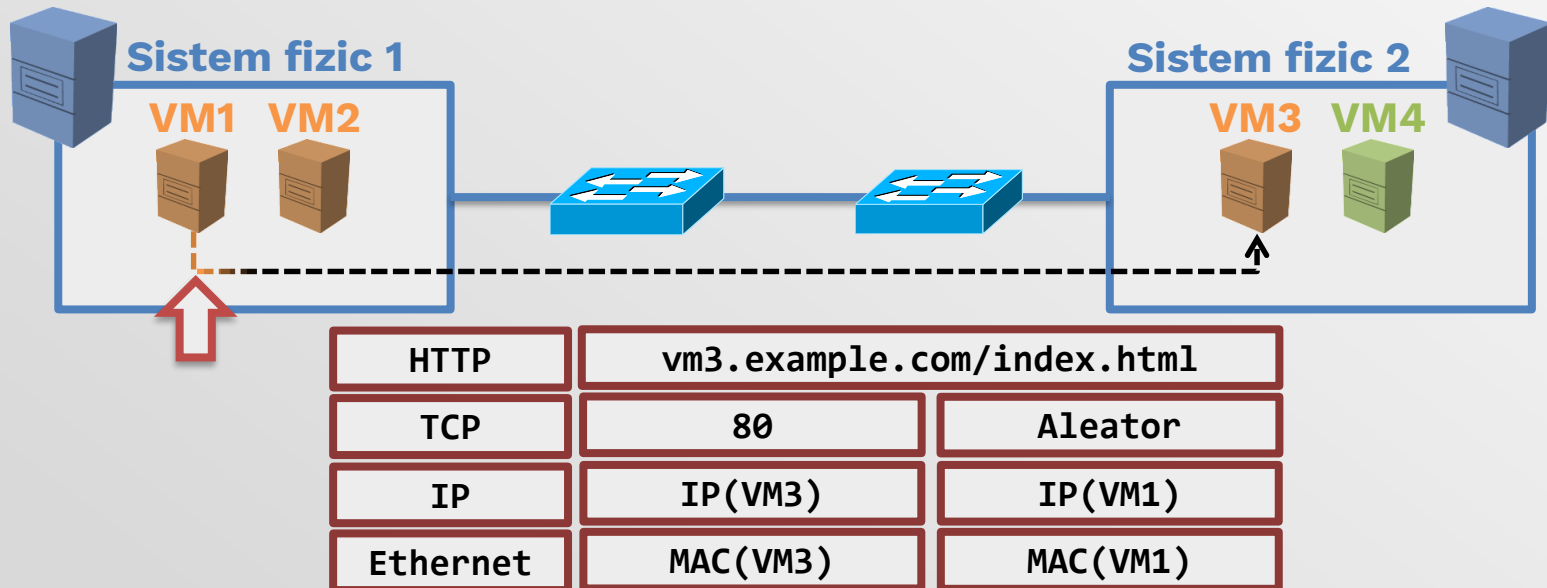
Tabela VXLAN

VNI = 42: VM3

VNI = 73: VM4

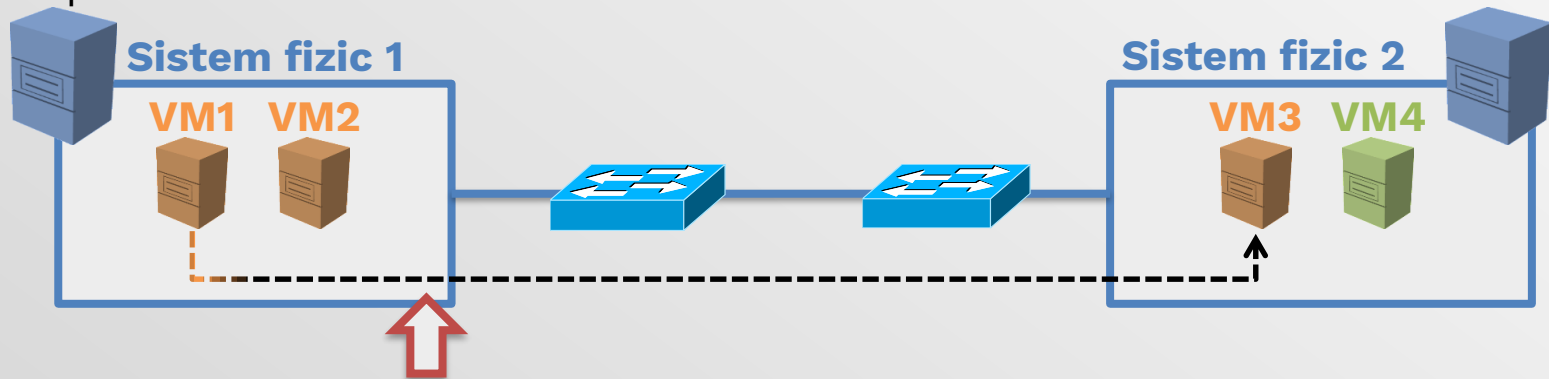
VXLAN

- **VM1** și **VM3** aparțin aceluiași client cloud (cu **VNI = 42**)
- **VM1** încearcă să recupereze prin HTTP o pagină de pe **VM3**
- **VM1** generează o cerere obișnuită HTTP (virtualizarea este transparentă)



VXLAN

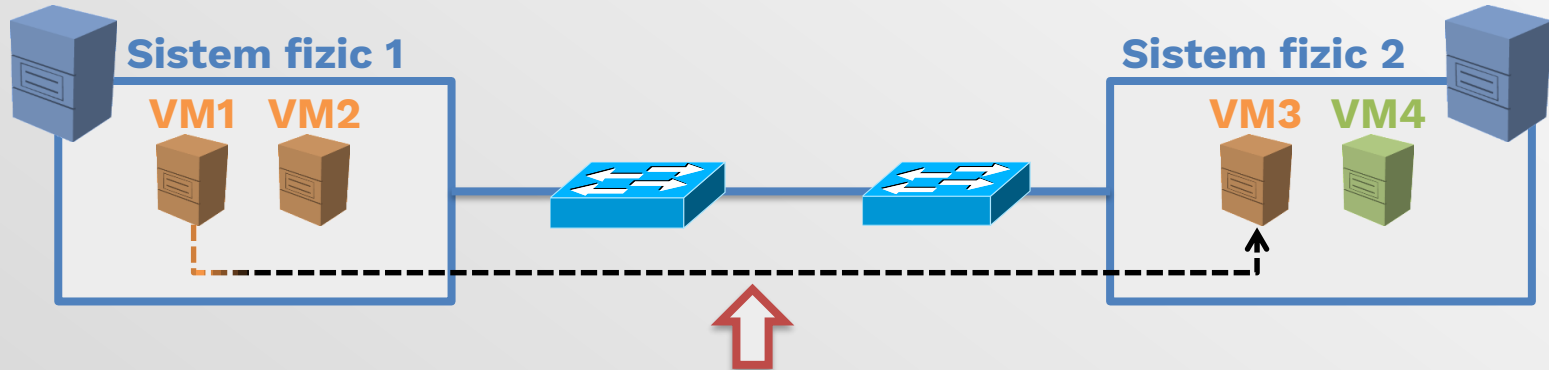
- La ieșirea din **VM1**, hypervisor-ul descoperă din ce rețea virtuală face parte traficul (codificată prin **VNI**)
- Tot pachetul este încapsulat într-un antet UDP, împreună cu informația despre **VNI**



HTTP	vm3.example.com/index.html	
TCP	80	Aleator
IP	IP(VM3)	IP(VM1)
Ethernet	MAC(VM3)	MAC(VM1)
VXLAN	VNI = 42	
UDP	8472	Aleator

VXLAN

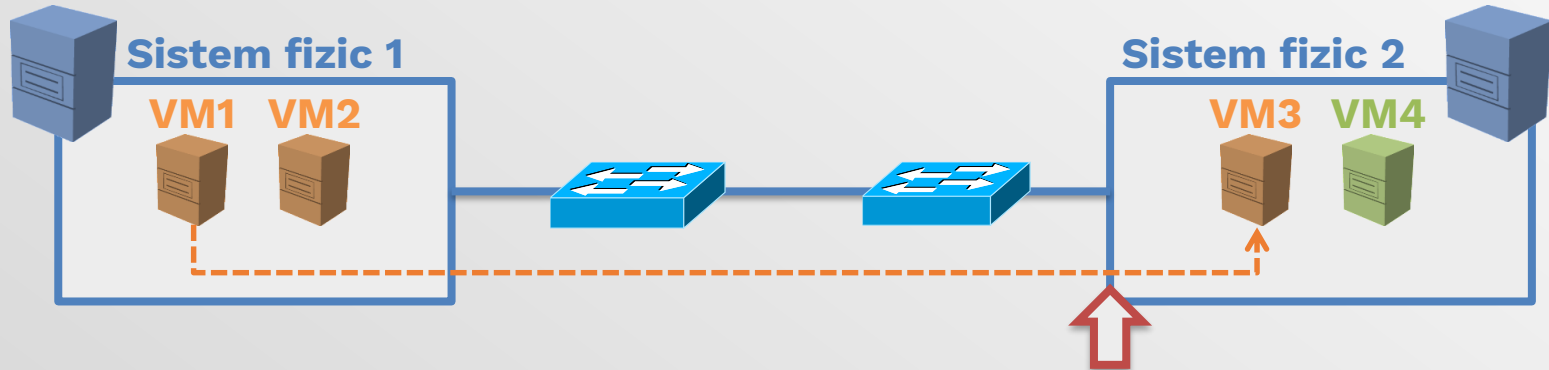
- La ieșirea din sistemul fizic, pachetul primește informații standard de nivel 2 și nivel 3 pentru a putea ajunge la mașina fizică destinație



HTTP	vm3.example.com/index.html	
TCP	80	Aleator
IP	IP(VM3)	IP(VM1)
Ethernet	MAC(VM3)	MAC(VM1)
VXLAN	VNI = 42	
UDP	8472	Aleator
IP	IP(Fizic2)	IP(Fizic1)
Ethernet	MAC(Fizic2)	MAC(Fizic1)

VXLAN

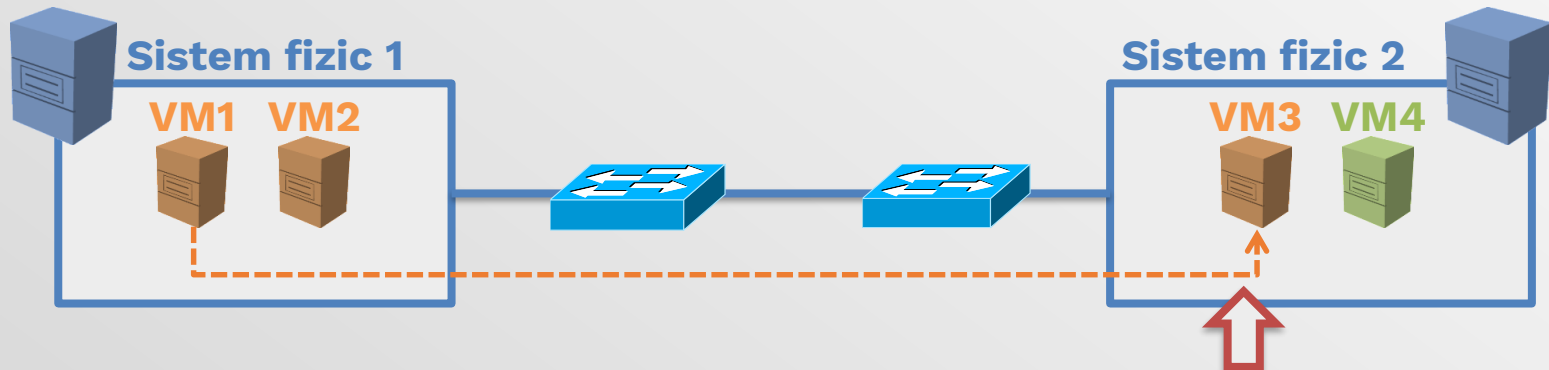
- Hypervisor-ul din sistemul destinație extrage informația de VNI și descoperă cărei mașini trebuie să-i trimită traficul



HTTP	vm3.example.com/index.html	
TCP	80	Aleator
IP	IP (VM3)	IP (VM1)
Ethernet	MAC (VM3)	MAC (VM1)
VXLAN	VNI = 42	
UDP	8472	Aleator

VXLAN

- La **VM3** pachetul ajunge ca un mesaj obișnuit HTTP
- Virtualizarea rețelei s-a făcut transparent din punctul de vedere al mașinilor virtuale



HTTP	vm3.example.com/index.html	
TCP	80	Aleator
IP	IP(VM3)	IP(VM1)
Ethernet	MAC(VM3)	MAC(VM1)

Soluția 2: NVGRE

- Actual draft IETF
- Soluție concurentă VXLAN
- Foarte similar ca mod de operare cu VXLAN
- Folosește încapsularea în pachete GRE în loc de încapsularea în segmente UDP
- Atât NVGRE cât și VXLAN sunt disponibile în implementarea de switch virtual **Open vSwitch**

Dispozitive virtuale în Linux

- Interfețe virtuale
- Switch-uri virtuale



RL
crunch it connected

Interfețe virtuale

- Sistemul de operare Linux permite crearea mai multor tipuri de interfețe de rețea virtuale

Tip interfață	Scop
Loopback	Testarea stivei de protocoale
Dummy	Testarea de configurări
Tun	Interfață de nivel 3 folosită de obicei pentru tunelare
Tap	Interfață de nivel 2 folosită de obicei pentru conectarea mașinilor virtuale KVM/QEMU
Bridge	Simulează un switch la care se pot conecta alte interfețe

- La acestea se adaugă interfețe virtuale caracteristice diferitelor soluții de virtualizare (VMware, VirtualBox, LXC, etc.)

Interfața de Loopback

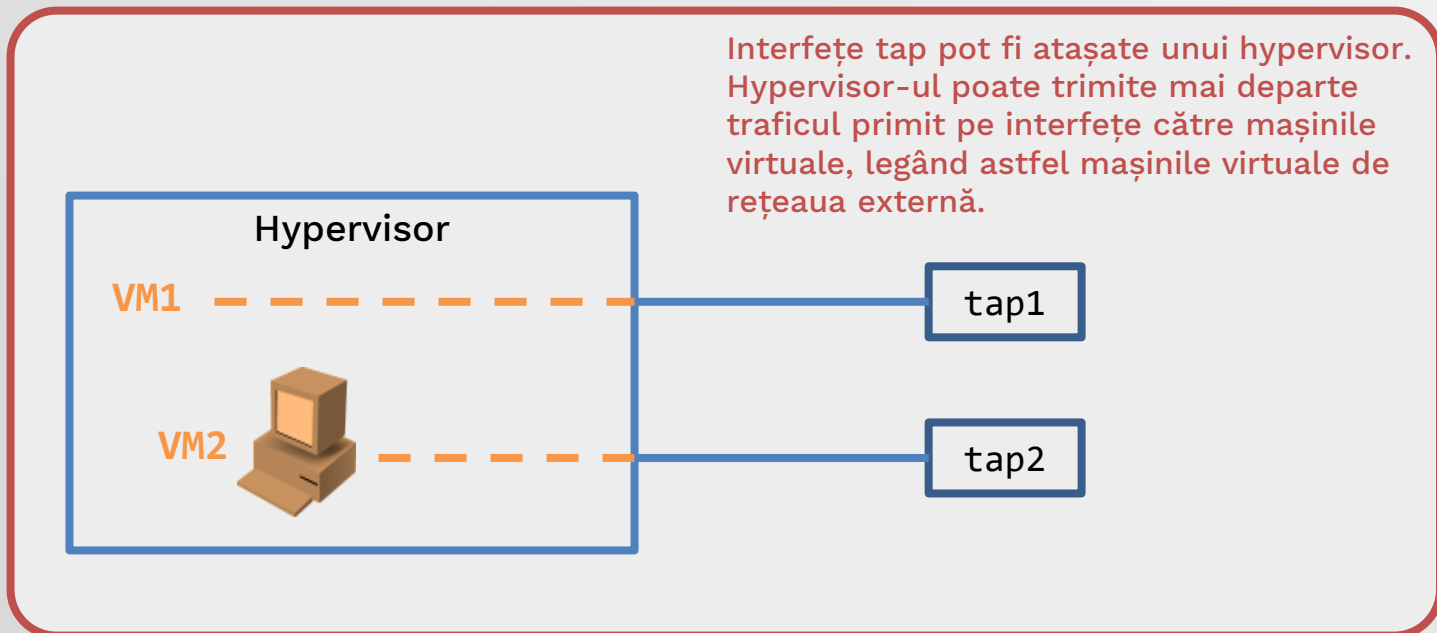
- Folosită pentru testarea stivei de protocoale
- Dacă ping în interfața de loopback eșuează, există probabil o defecțiune la nivel de sistem în protocolul IP

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
```

```
There's no place
like 127.0.0.1
```

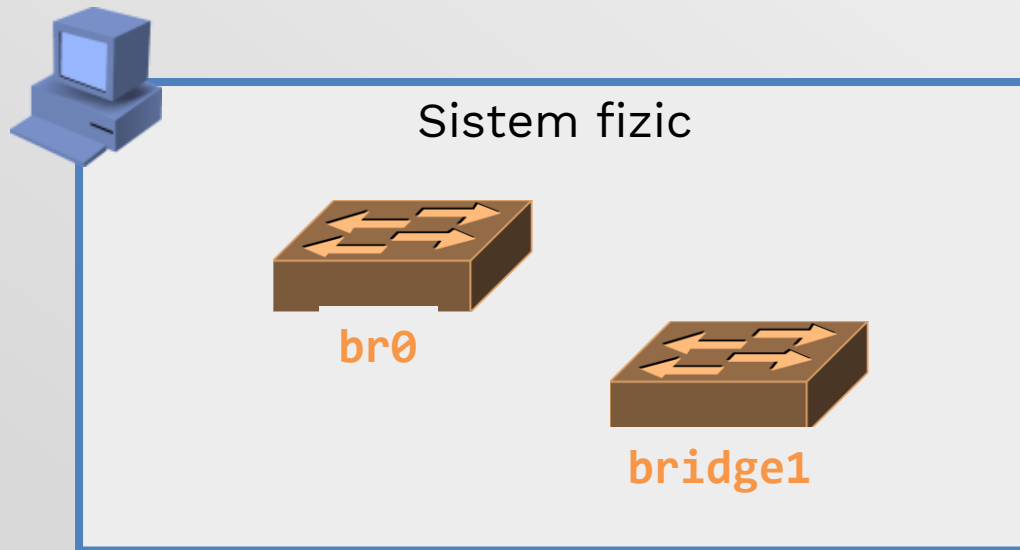
Interfețe tun/tap

- Interfețe similare ca scop
 - Tun folosește trafic de nivel 3
 - Tap folosește trafic de nivel 2
- Aplicații pot fi atașate de interfață
- Tot traficul trimis pe interfață este trimis de fapt aplicației



Switch-uri virtuale

- Permit interconectarea interfețelor de pe un sistem
- De obicei create cu numele **br0**, **br1**, **bridge0**, **bridge1**, etc.
- Dispun de funcționalități de VLAN și STP
- Instalate prin pachetul bridge-utils



Switch-uri virtuale

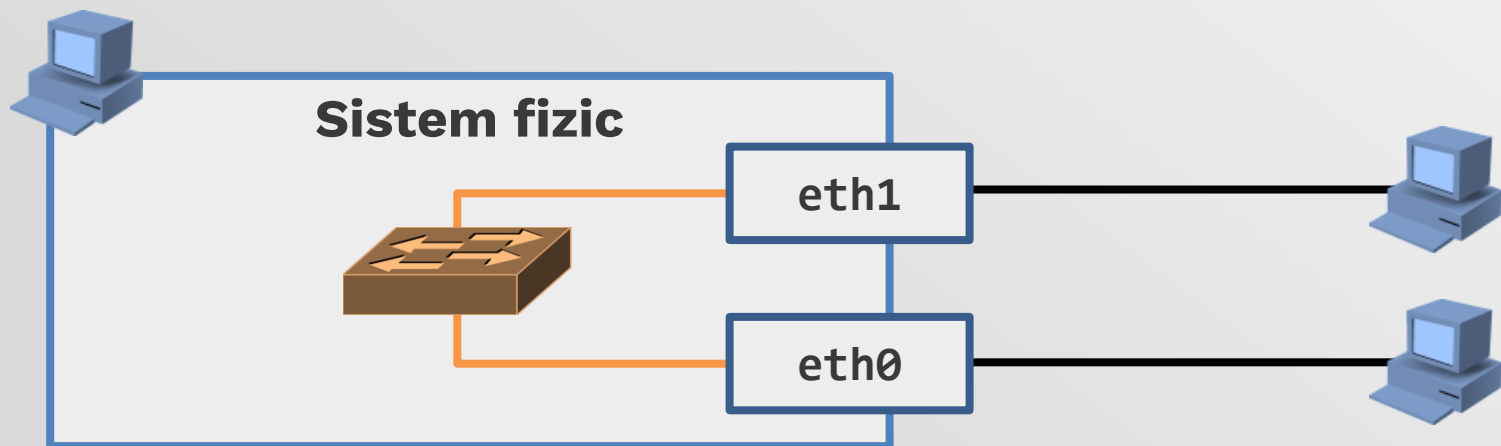
- Pentru a crea un switch virtual nou:

```
linux# brctl addbr br0
```

- Pentru a adăuga interfețe la switch-ul virtual:

```
linux# brctl addif br0 eth0
```

```
linux# brctl addif br0 eth1
```



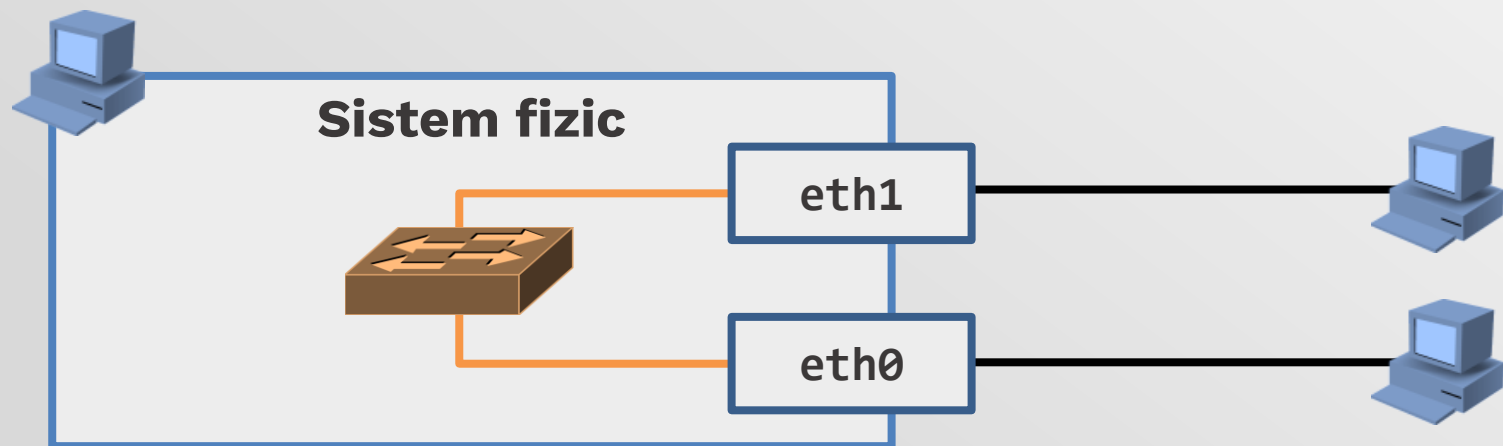
Switch-uri virtuale

- Switch-ul virtual poate fi oprit sau pornit ca orice altă interfață:

```
linux# ip link set dev br0 up
```

- Pentru a adăuga interfețe la switch-ul virtual:

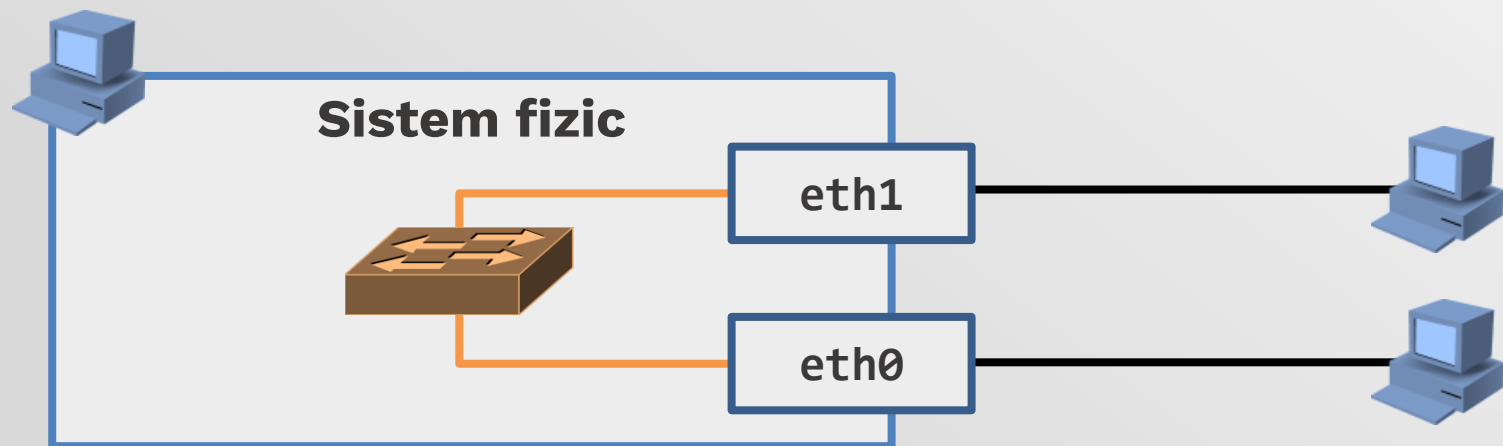
```
linux# brctl show
```



Switch-uri virtuale

- Pentru a șterge interfețe din switch:

```
linux# brctl delif br0 eth0
```



Switch-uri virtuale cu VLAN-uri

- Linux permite lucrul cu VLAN-uri
- Trafic 802.1Q poate fi trimis în rețeaua externă, dacă placa de rețea este compatibilă 802.1Q
- Comanda pentru lucrul cu VLAN-uri este **vconfig**

```
linux# vconfig add [interface-name] [vlan-id]
```

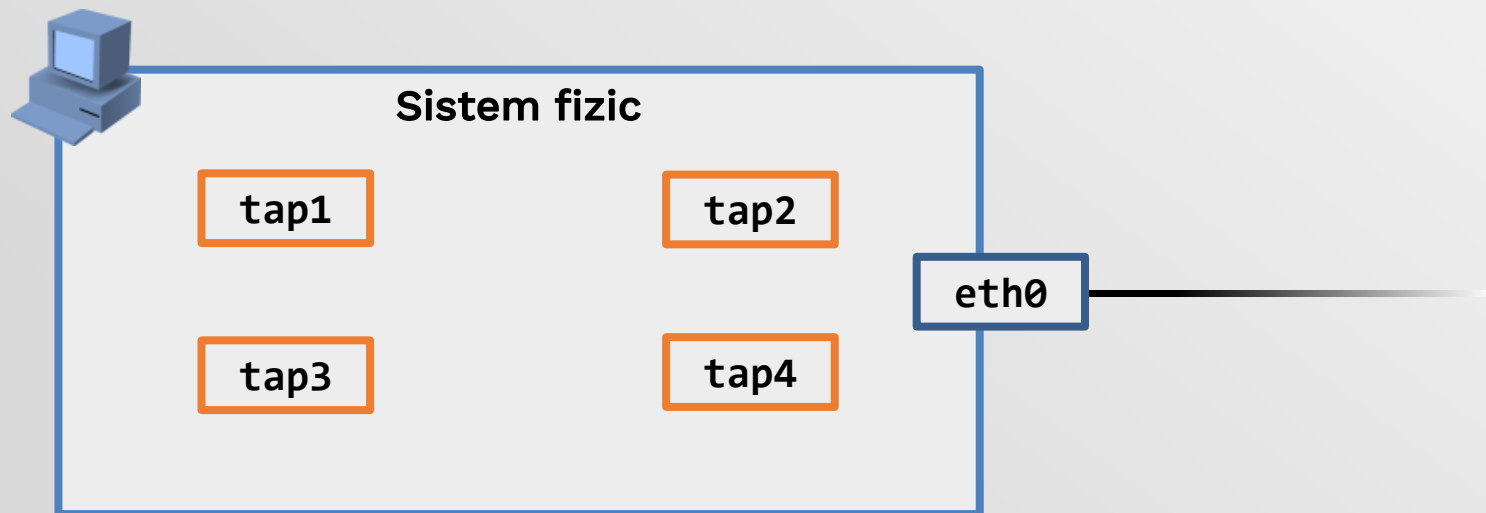
- Comanda creează o subinterfață de VLAN, ce va trimite trafic 802.1Q

```
linux# vconfig add eth0 10 → eth0.10@eth0
```

```
linux# vconfig add eth0 20 → eth0.20@eth0
```

Switch-uri virtuale cu VLAN-uri

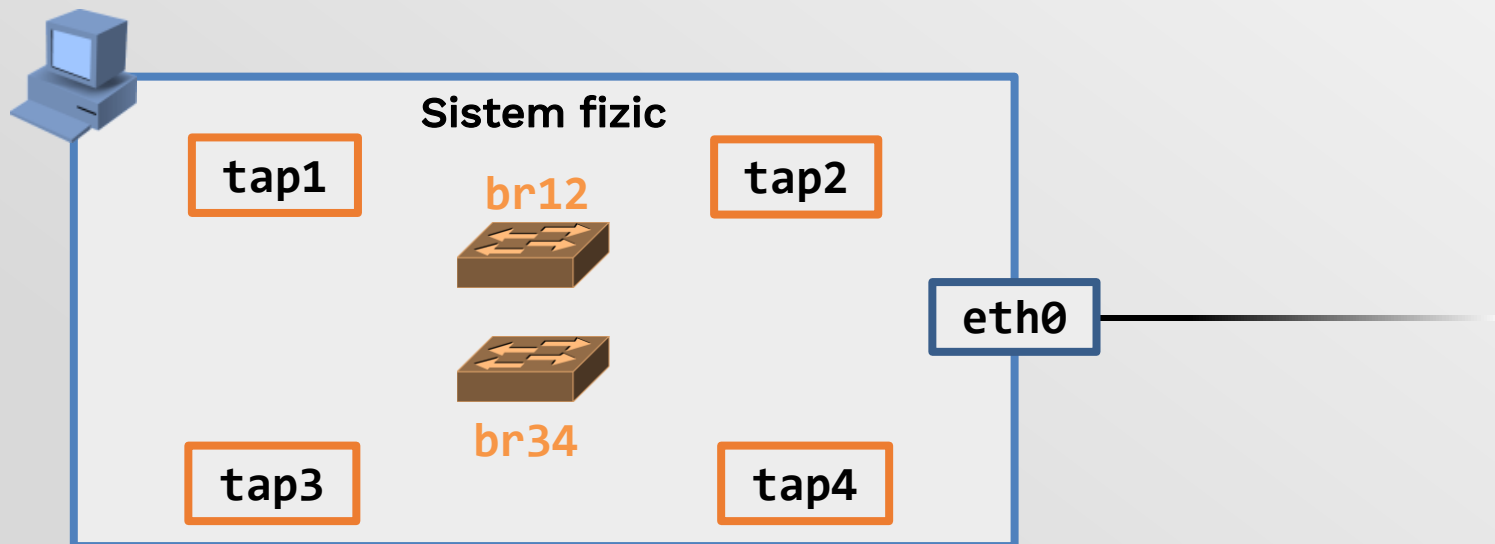
- În scenariul de mai jos, interfețele virtuale tap sunt atașate la patru mașini virtuale (**tap1** la **VM1**, **tap2** la **VM2**, ș.a.m.d.)
- Vrem să avem conectivitate între mașinile virtuale și **eth0**, fără ca acestea să poată comunica între ele



Switch-uri virtuale cu VLAN-uri

- Creăm două switch-uri virtuale

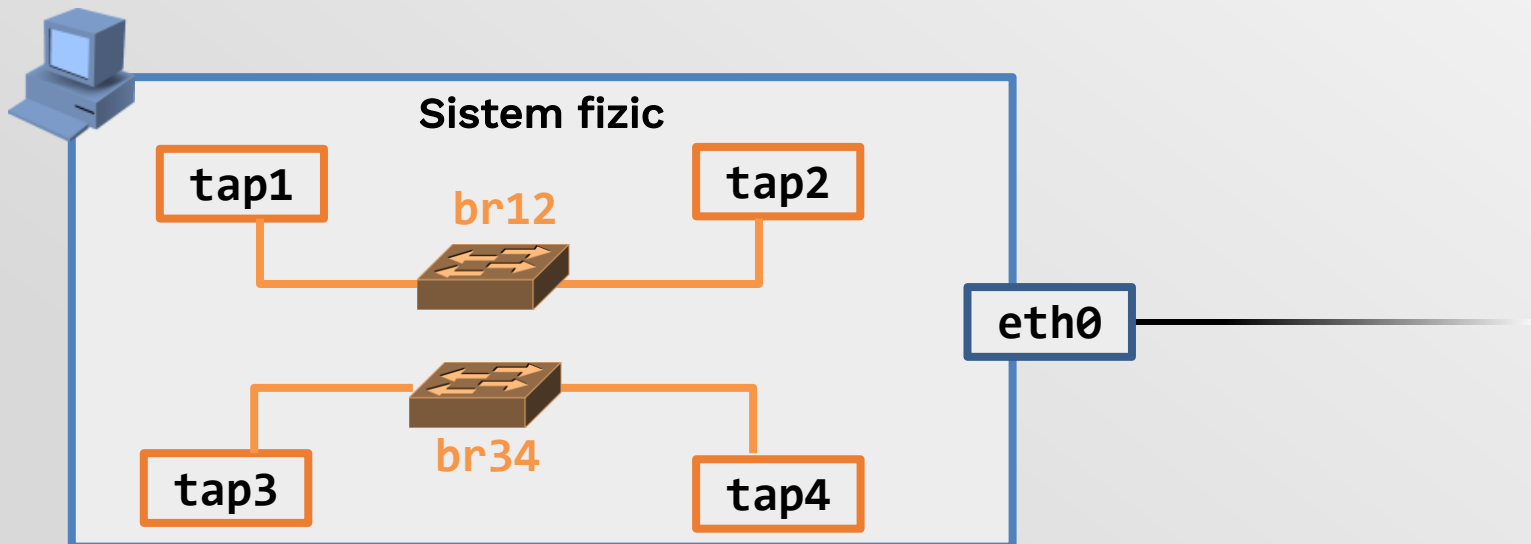
```
linux# brctl addbr br12
linux# brctl addbr br34
linux# ip link set dev br12 up
linux# ip link set dev br34 up
```



Switch-uri virtuale cu VLAN-uri

- Atașăm mașinile virtuale la switch-urile create

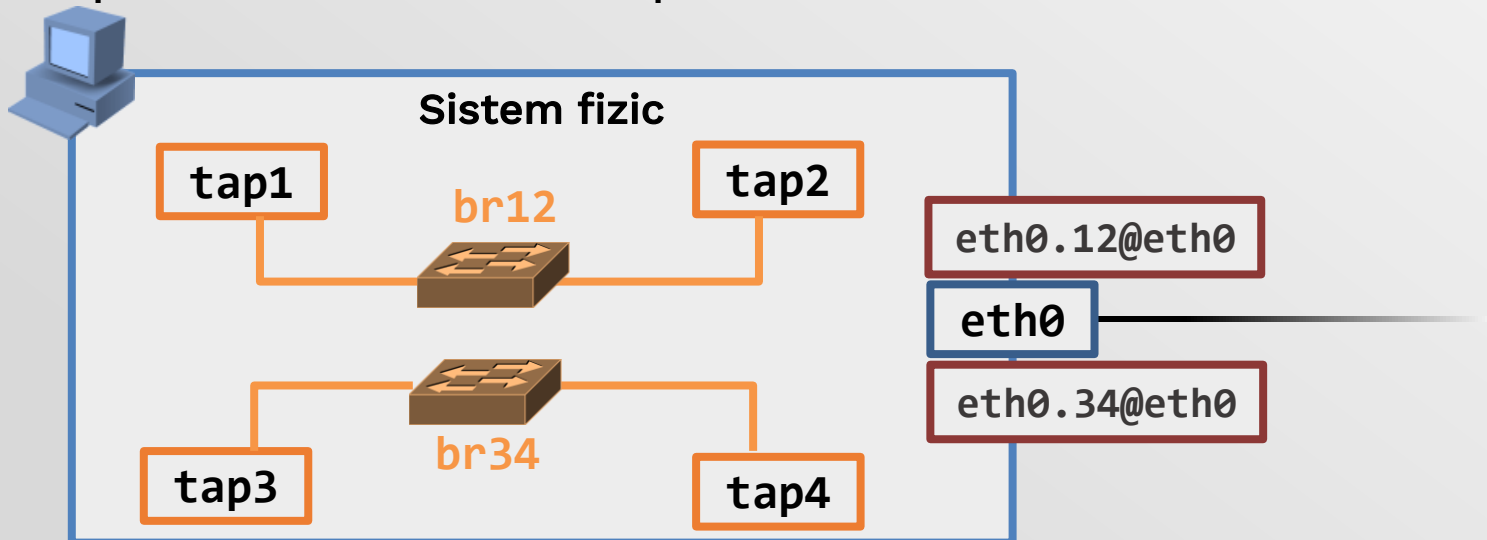
```
linux# brctl addif br12 tap1
linux# brctl addif br12 tap2
linux# brctl addif br34 tap3
linux# brctl addif br34 tap4
```



Switch-uri virtuale cu VLAN-uri

- Vrem ca **tap1** și **tap2** să fie în VLAN12, și **tap3** și **tap4** în VLAN34
- Creăm subinterfețe din fiecare VLAN pentru **eth0**

```
linux# vconfig add eth0 12
linux# vconfig add eth0 34
linux# ip link set dev eth0.12 up
linux# ip link set dev eth0.34 up
```

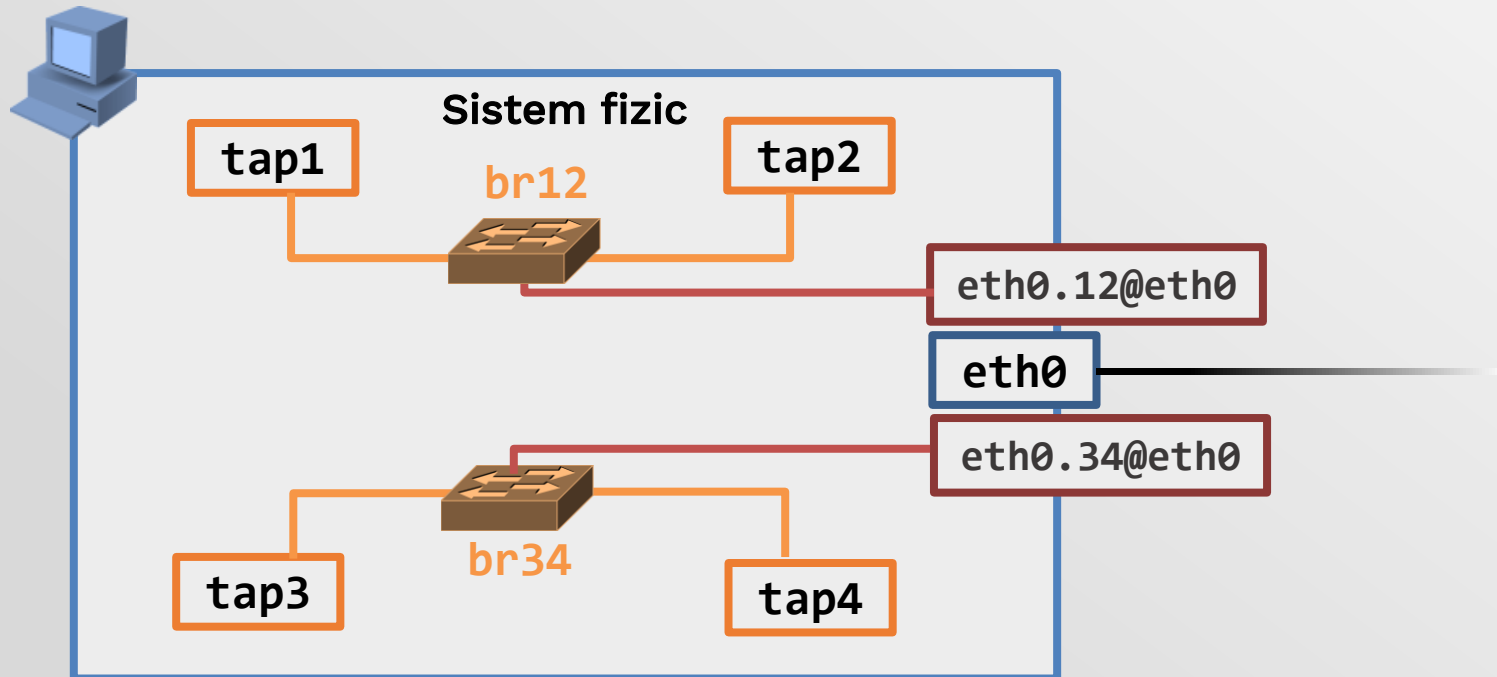


Switch-uri virtuale cu VLAN-uri

- Conectăm subinterfețele la switch-uri

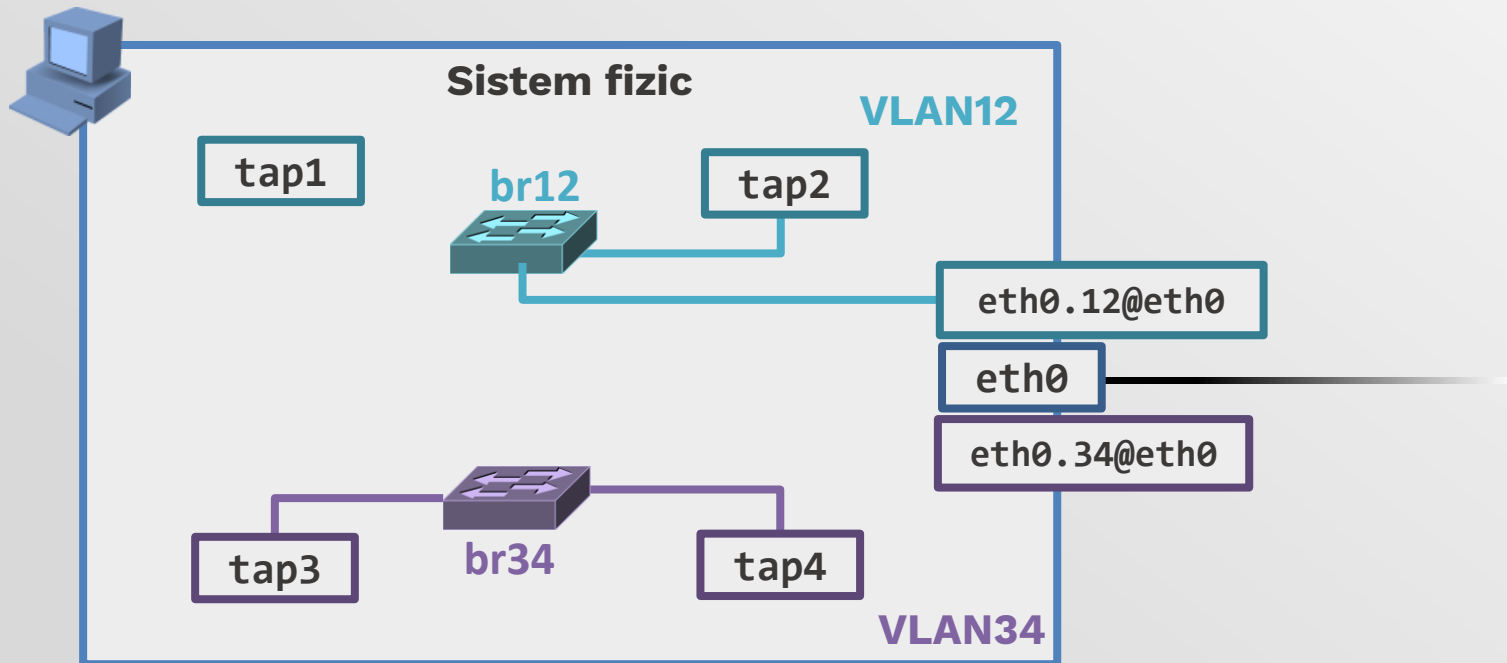
```
linux# brctl addif br12 eth0.12
```

```
linux# brctl addif br34 eth0.34
```



Switch-uri virtuale cu VLAN-uri

- **VM1** și **VM2** pot comunica prin **eth0** (via **eth0.12**)
- **VM3** și **VM4** pot comunica prin **eth0** (via **eth0.34**)
- **VM1/VM2** nu pot comunica cu **VM3/VM4**



VMware

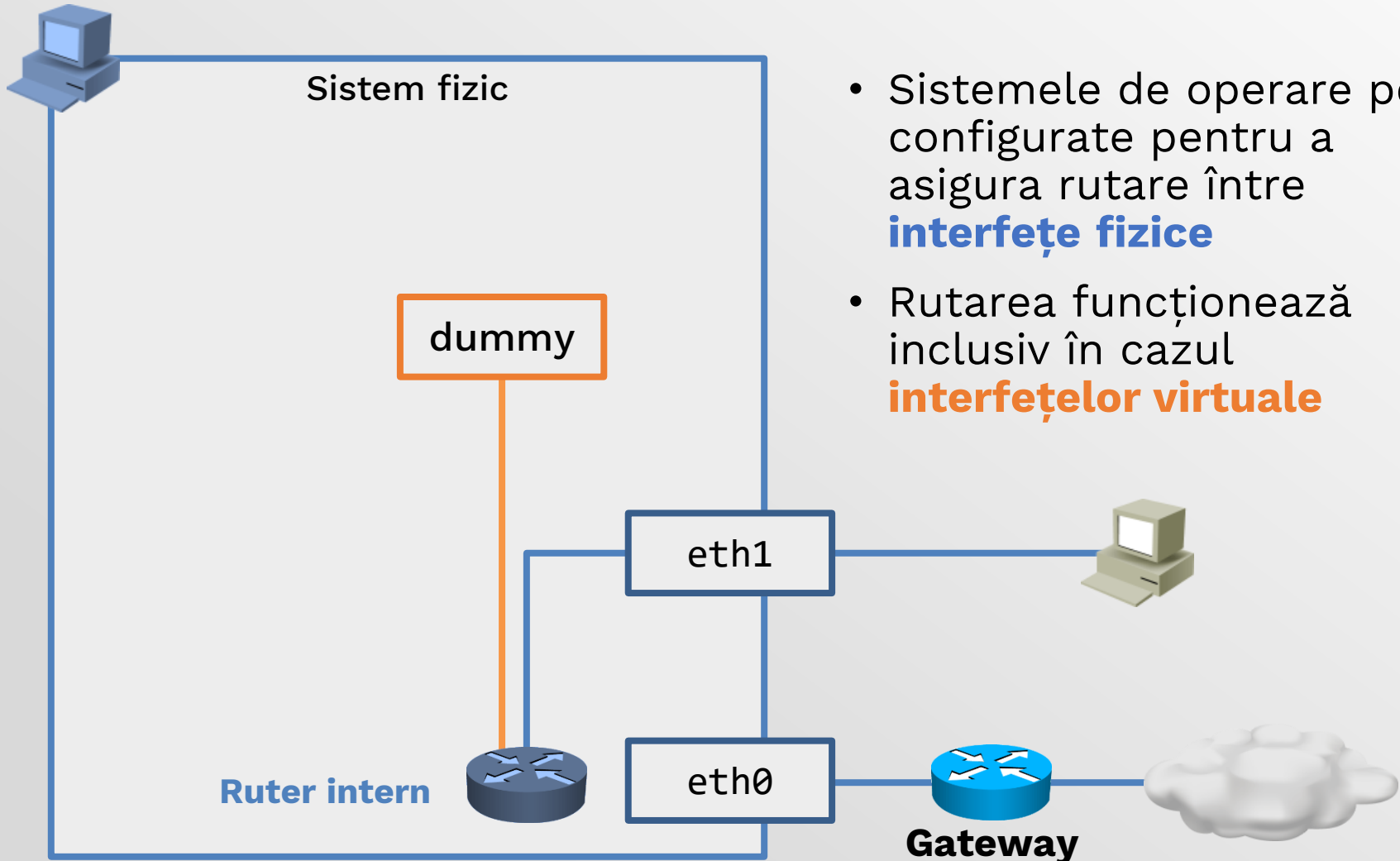
- Descriere
- Rețele cu bridging
- Rețele cu NAT
- Rețele host-only



VMware

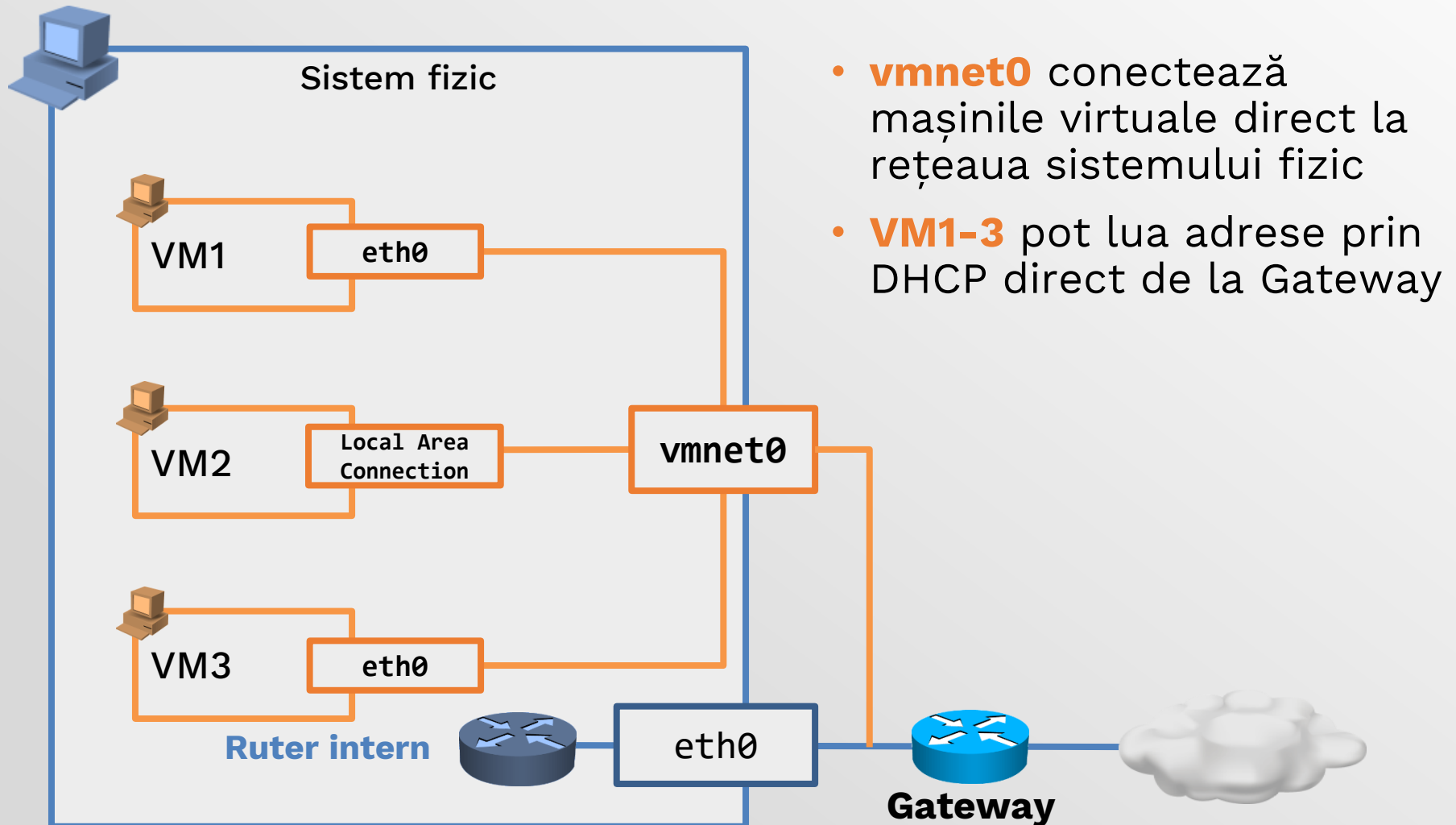
- VMware Player / VMware Workstation
- Pentru a lega mașinile virtuale dispune de mai multe switch-uri virtuale (numite **vmnet**)
- Switch-urile virtuale permit conectarea în mai multe moduri:
 - **NAT**: mașina fizică va face NAT pe traficul mașinilor virtuale
 - **Host-only**: mașinile virtuale sunt conectate la gazdă prin intermediul unei rețele situate în spatele acesteia
 - **Bridge**: interfețele mașinilor virtuale sunt atașate direct mediului fizic

Rutare în Linux



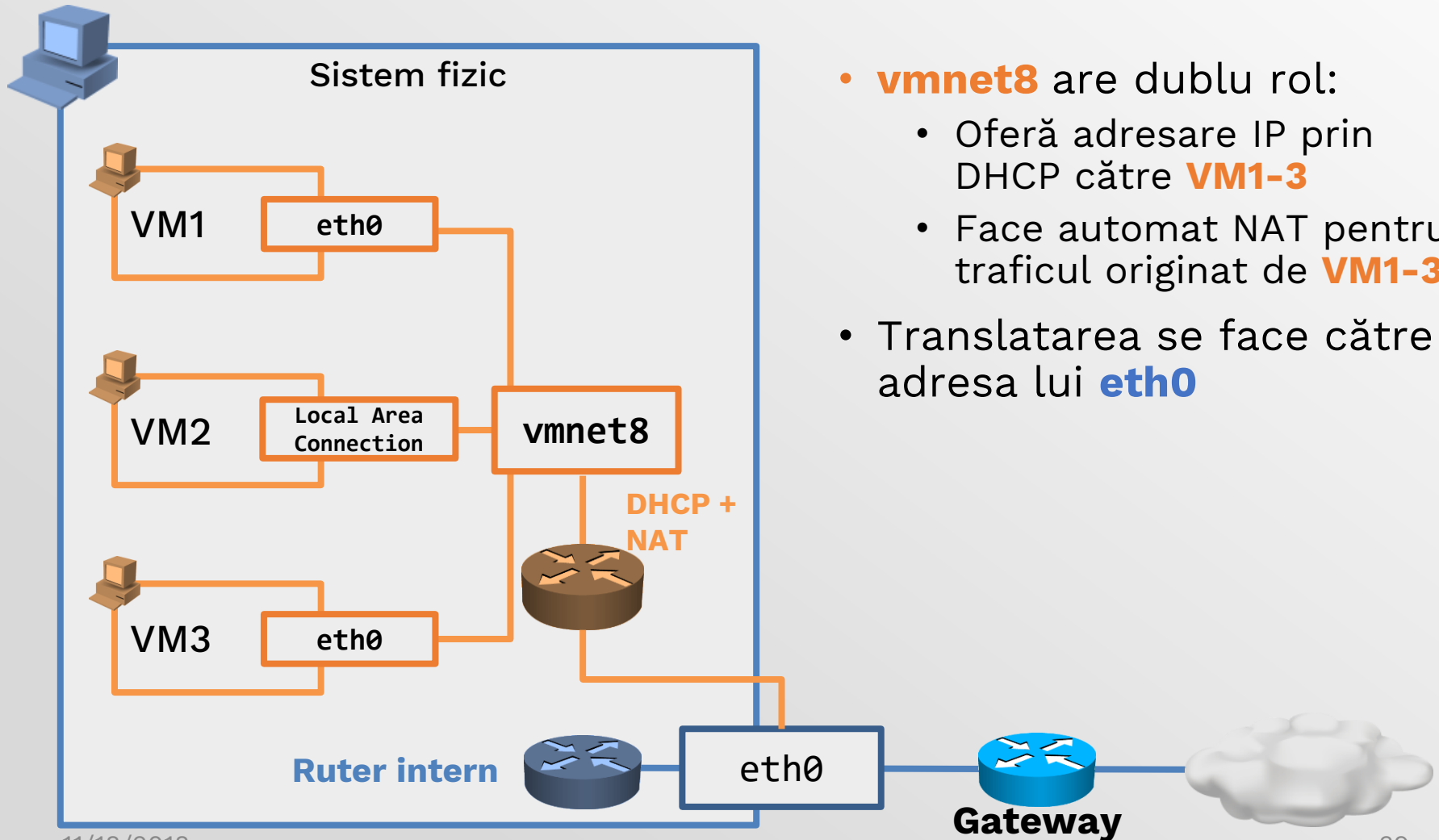
- Sistemele de operare pot fi configurate pentru a asigura rutare între **interfețe fizice**
- Rutarea funcționează inclusiv în cazul **interfețelor virtuale**

VMware: Bridged network



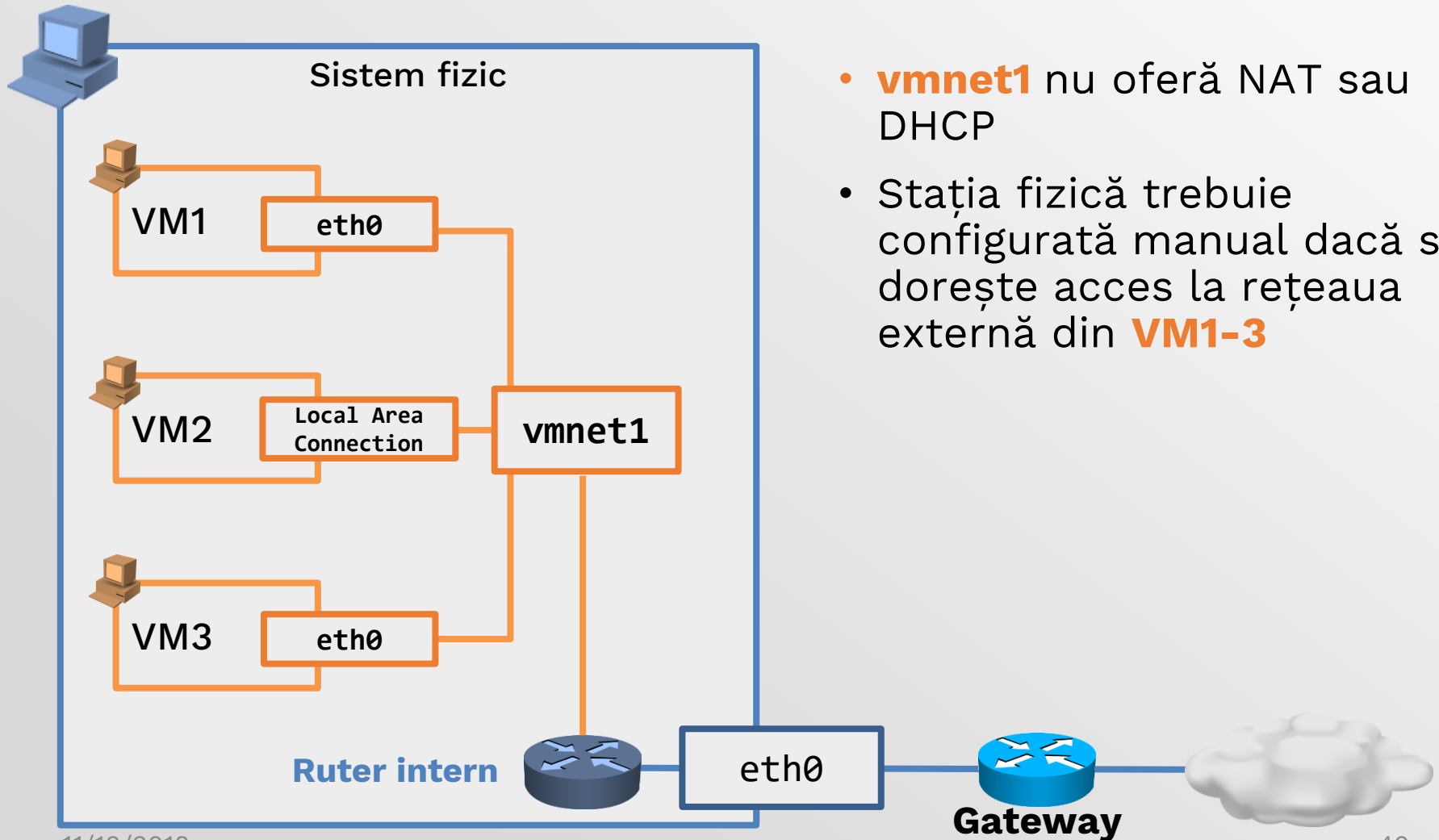
- **vmnet0** conectează mașinile virtuale direct la rețeaua sistemului fizic
- **VM1-3** pot lua adrese prin DHCP direct de la Gateway

VMware: NAT network



- **vmnet8** are dublu rol:
 - Oferă adresare IP prin DHCP către **VM1-3**
 - Face automat NAT pentru traficul originat de **VM1-3**
- Translatarea se face către adresa lui **eth0**

VMware: Host only network



- **vmnet1** nu oferă NAT sau DHCP
- Stația fizică trebuie configurată manual dacă se dorește acces la rețeaua externă din **VM1-3**

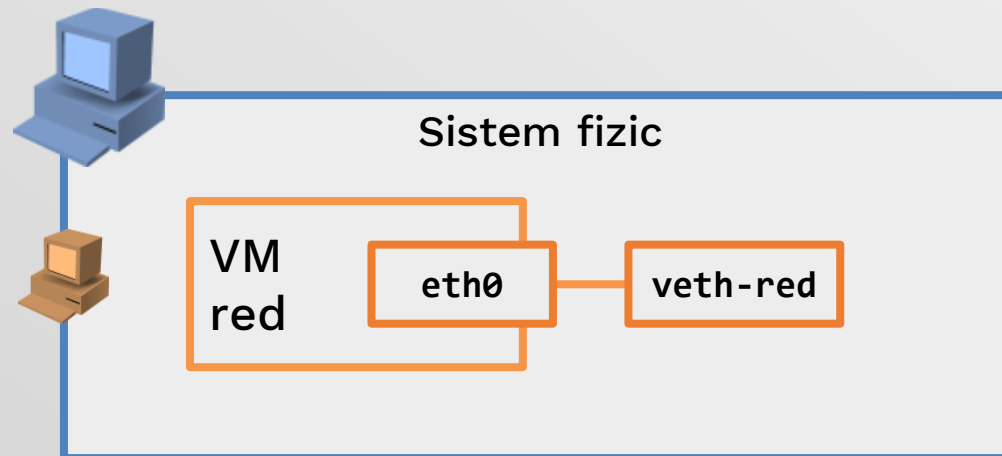
LXC

- Descriere
- Arhitectură
- Topologia de la laborator

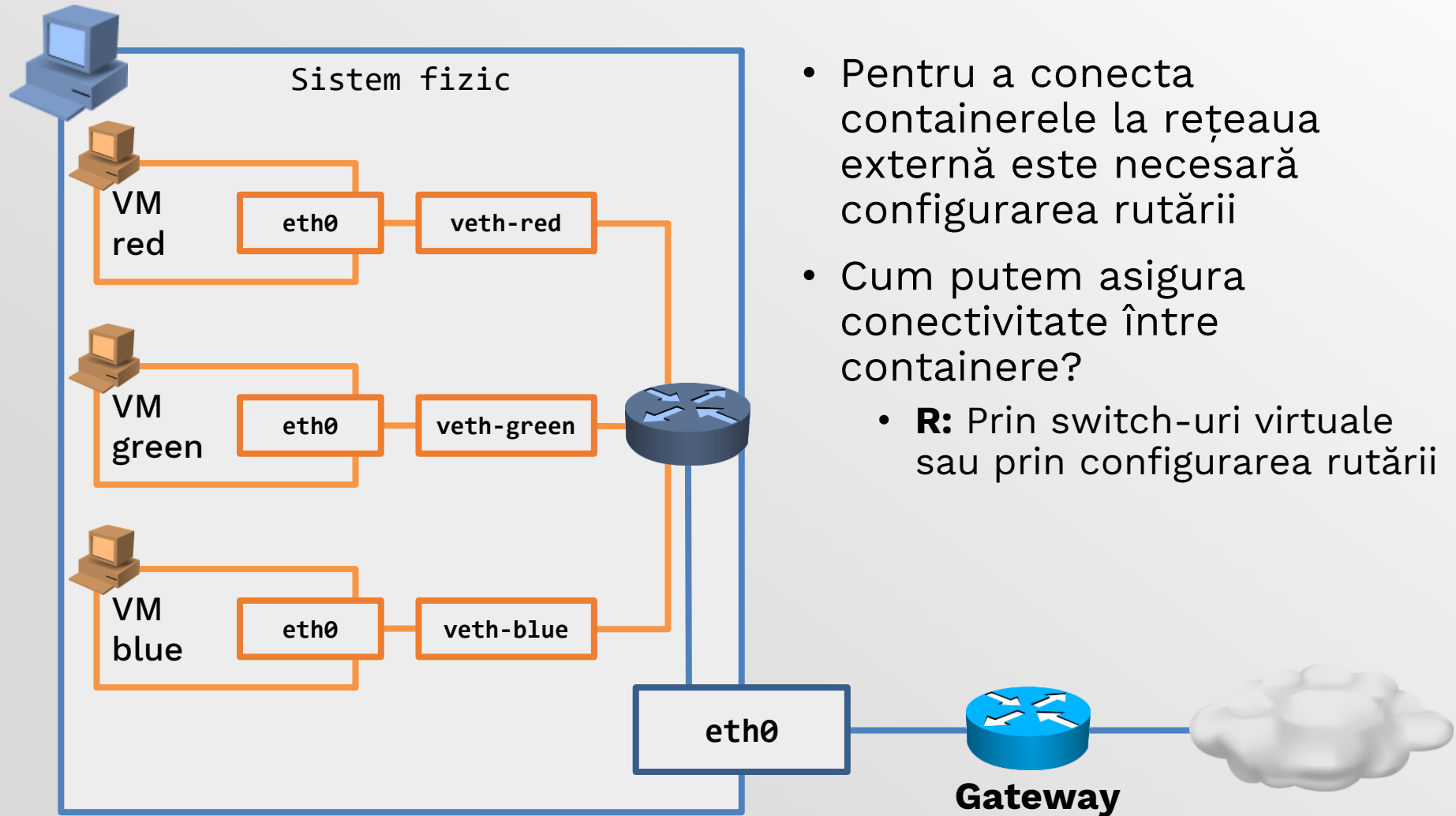


LXC

- Linux Containers
- Mașinile virtuale împart același kernel cu mașina fizică
- Mașinile virtuale LXC sunt de obicei numite **containere**
- Legătura dintre un container și sistemul gazdă se face prin interfețe virtuale de tip **veth**, atașate la interfețele **eth0** din container

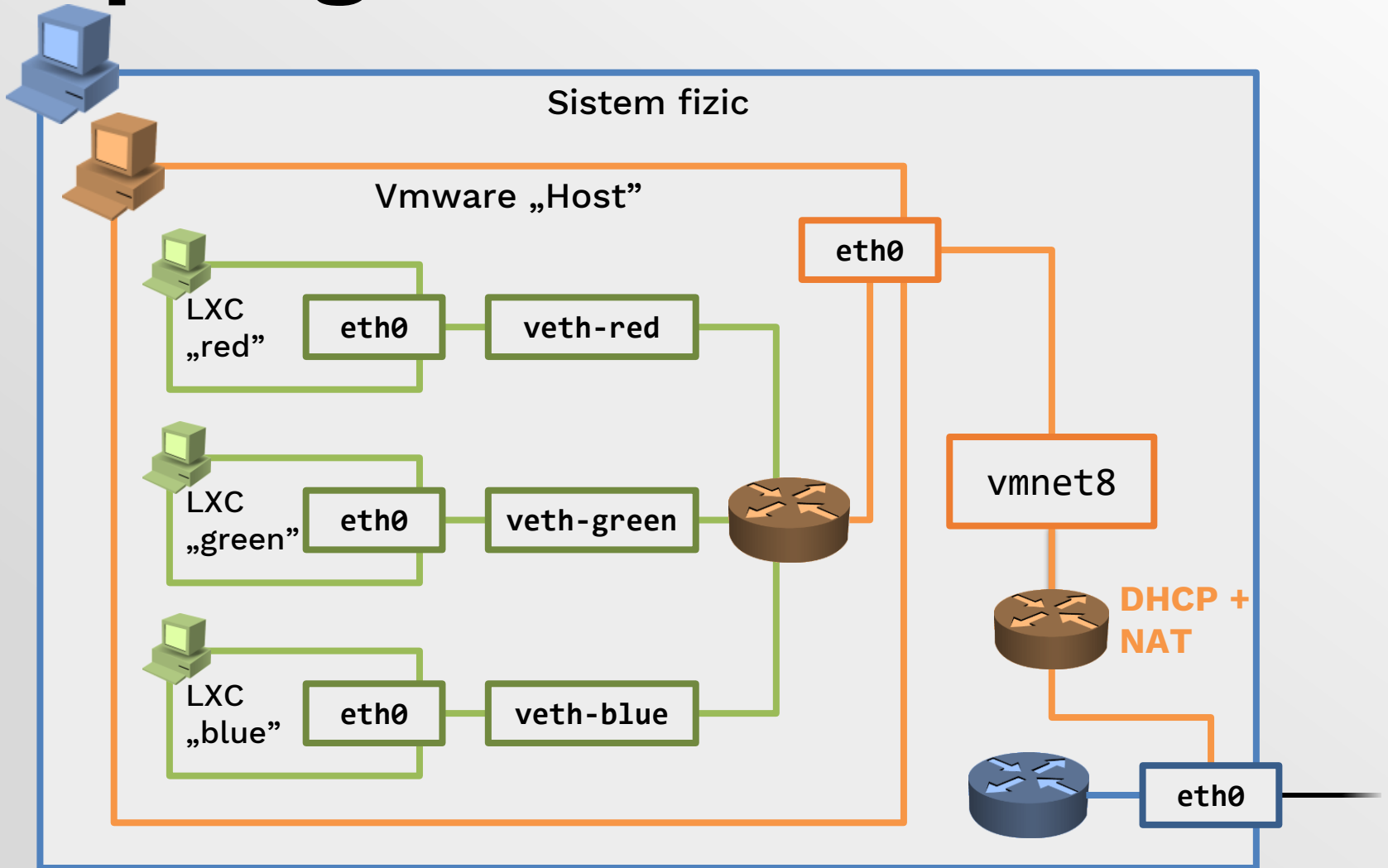


LXC



- Pentru a conecta containerele la rețeaua externă este necesară configurarea rutării
- Cum putem asigura conectivitate între containere?
 - **R:** Prin switch-uri virtuale sau prin configurarea rutării

Topologia de la laborator



SDN

- Definiție
- OpenFlow



RL
crunch it connected

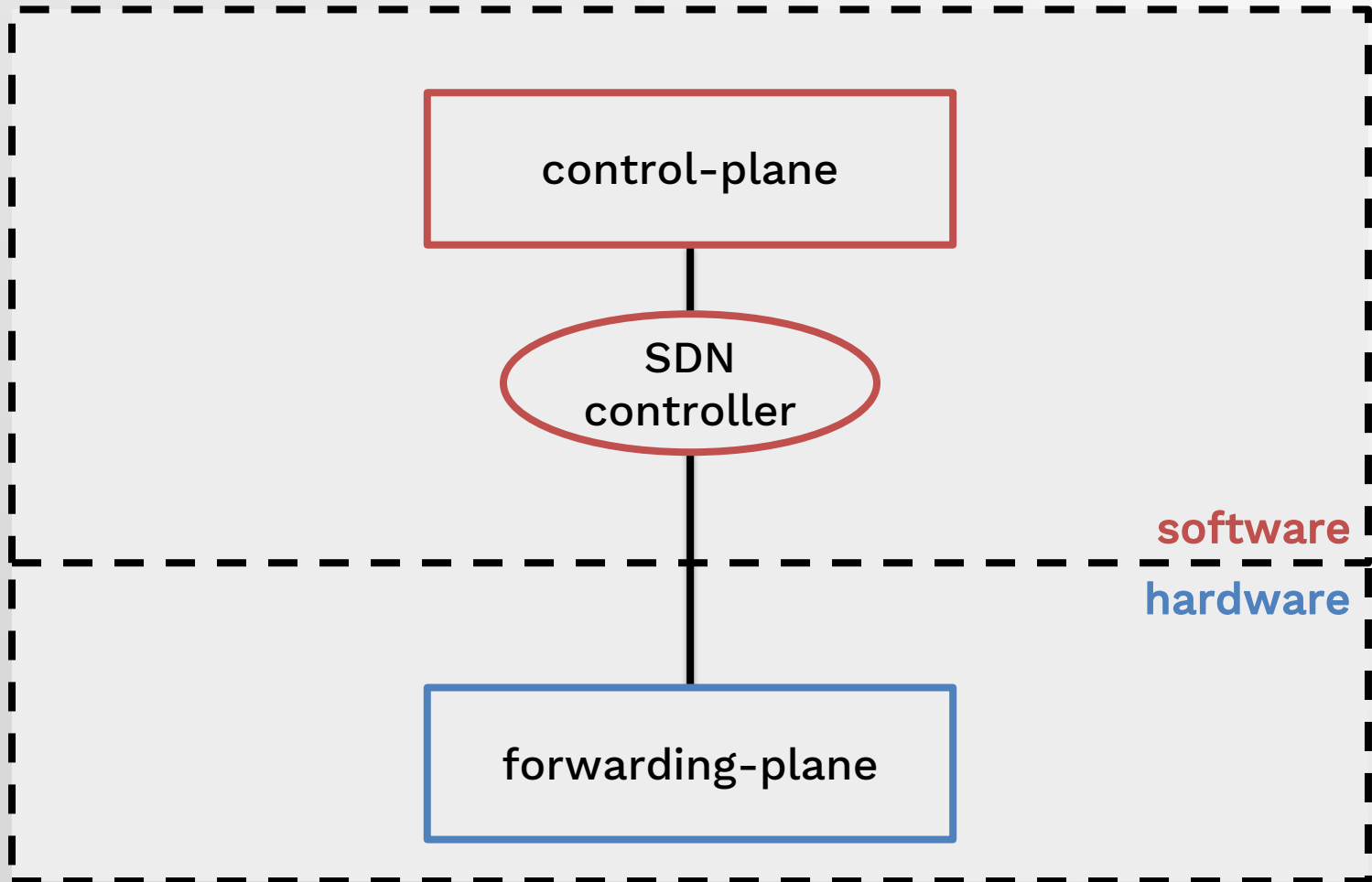
SDN

- Software Defined Networks – direcție nouă în dezvoltarea rețelelor
- Operațiunile realizate de echipamentele de rețea sunt împărțite în două aspecte:

Control-plane	<ul style="list-style-type: none"> • Componentele care decid cum să fie gestionat traficul (componenta decizională)
Forwarding-plane Data-plane	<ul style="list-style-type: none"> • Componentele care gestionează efectiv traficul (componenta executivă)

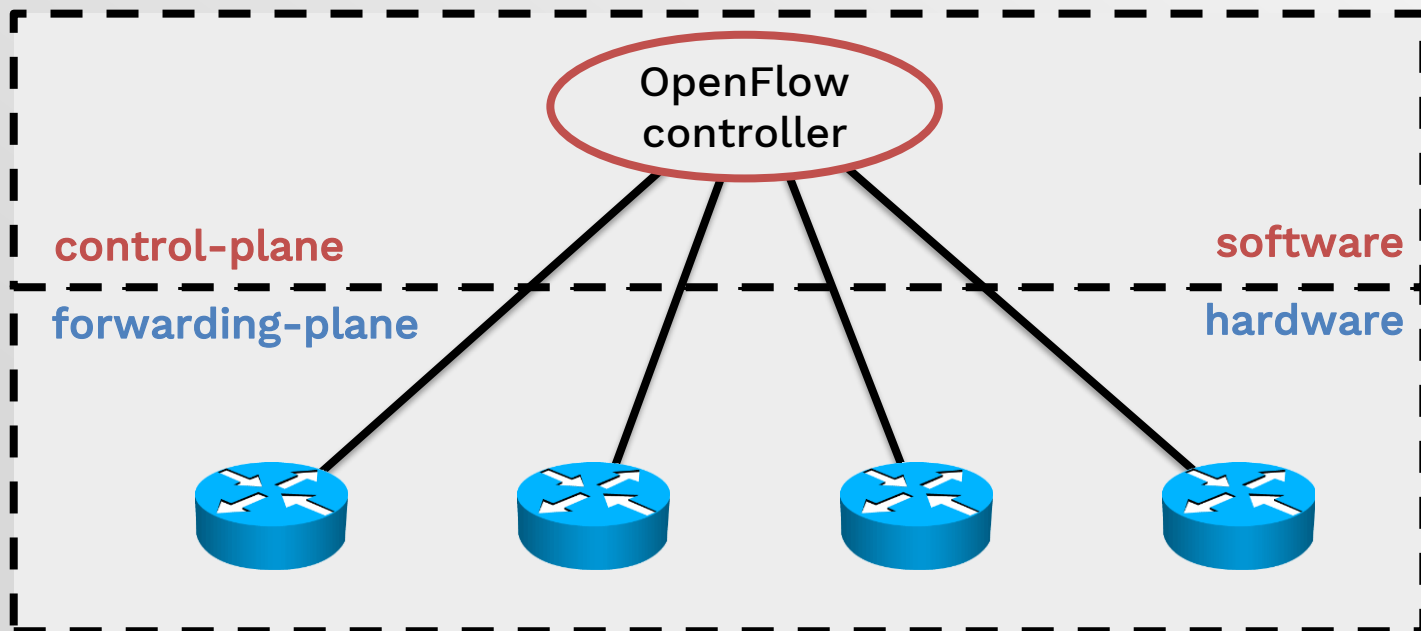
- SDN oferă utilizatorului o viziune centralizată, programatică, a aspectelor legate de control-plane
- Obiectiv: programatorul specifică printr-un limbaj ce dorește de la rețea, urmând ca aceasta să se ocupe automat de gestionarea efectivă a traficului conform specificațiilor

SDN

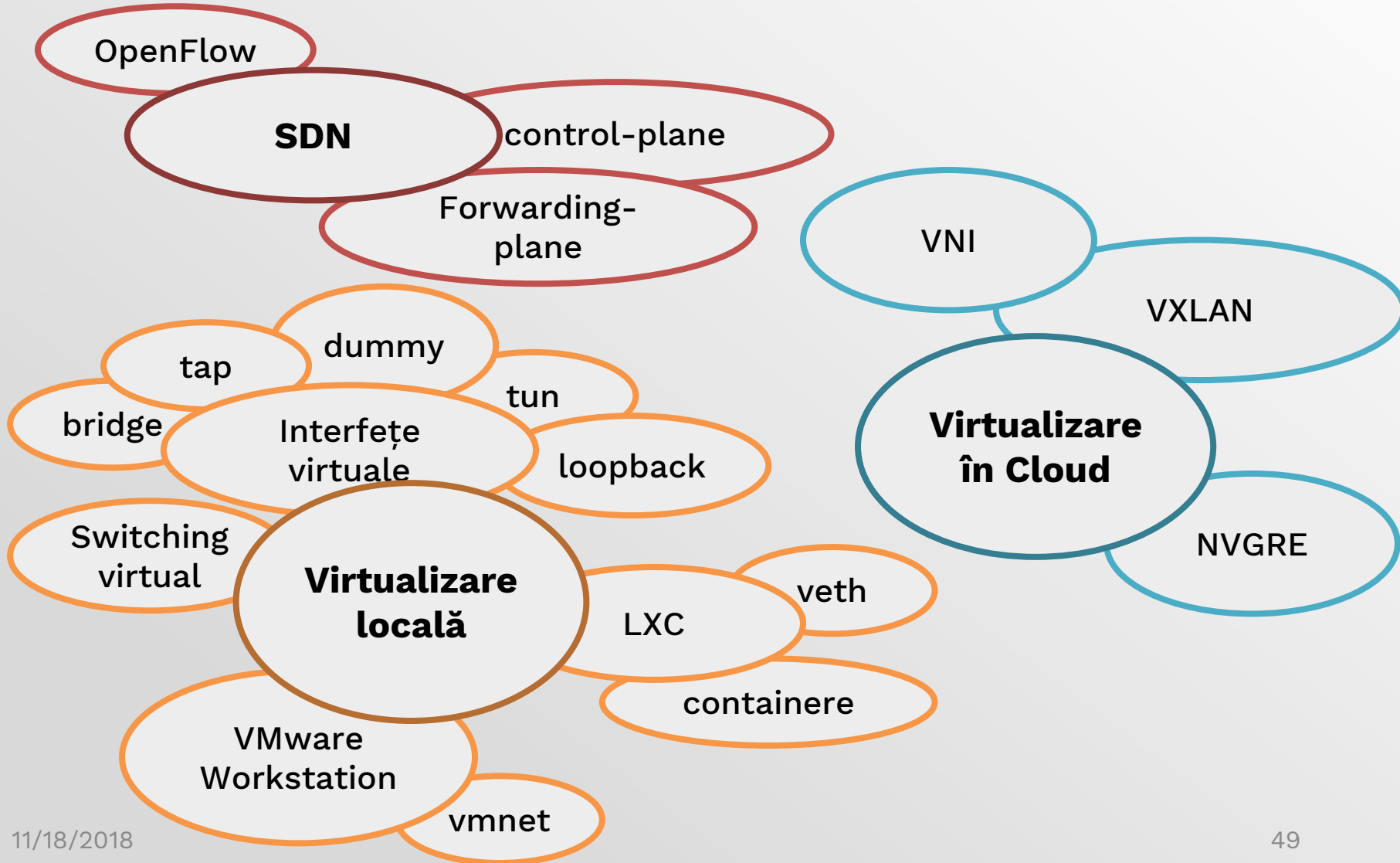


OpenFlow

- Protocol care permite accesul, prin rețea, la **forwarding-plane**-ul unui echipament de rețea
- **Control-plane**-ul este poziționat în afara dispozitivului, deciziile fiind luate de către un controller extern configurat de utilizator



Cuvinte cheie



Echipa de RL vă urează Vacanță plăcută!

