Virtualisation et réseau

Lucas Nussbaum
lucas.nussbaum@univ-lorraine.fr

Licence professionnelle ASRALL

Administration de systèmes, réseaux et applications à base de logiciels libres

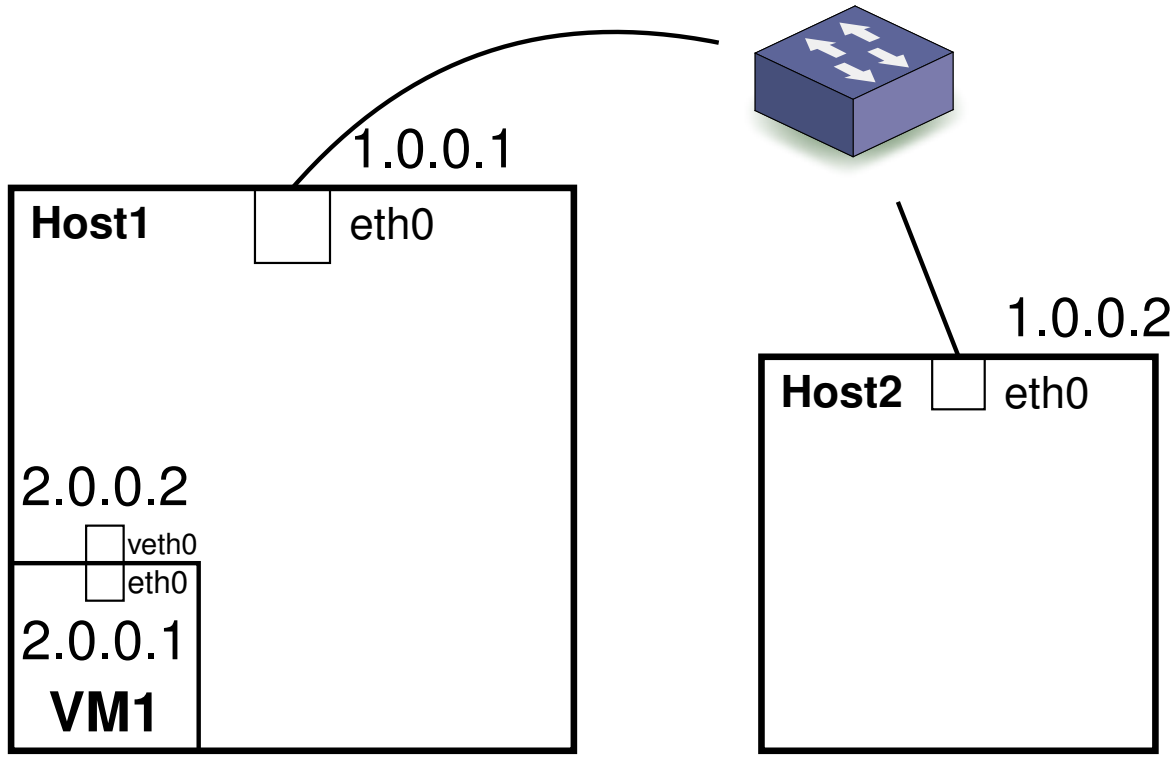
Introduction

- Sauf cas très particuliers, les VM ont besoin de communiquer par le réseau
- Différents modèles sont possibles :
  - L2 (bridging), L3 (routage), NAT, réseau virtuel (VXLAN), etc.
- Réseau L2 ($\approx$ réseau *ethernet*)
  - Exemple : salle de TP, ou ensemble de salles de TP
  - Toutes les machines voient les broadcasts de toutes les autres
  - Même plage d'adresses pour les machines (152.81.5.0/24)
  - ARP : savoir à quelle adresse MAC correspond une adresse IP
- Routage
  - Pour interconnecter des réseaux L2
  - Routeurs (équipements réseaux ou machines)
  - Tables de routage pour orienter les paquets

Routage avec Linux

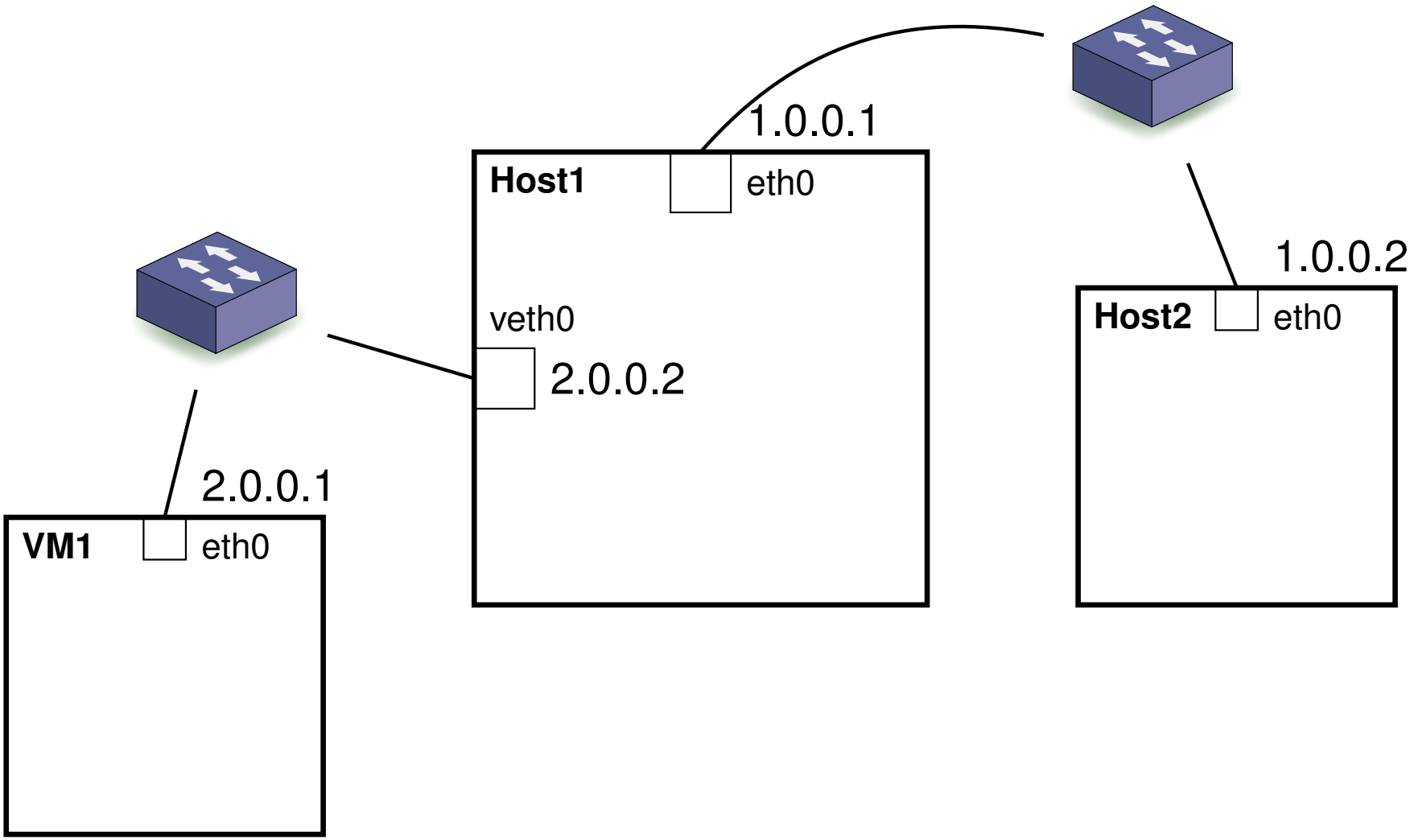
- Linux peut servir de routeur, mais il faut l'activer :
  `echo 1 > /proc/sys/net/ipv4/ip_forward`
  (vérifier aussi les règles éventuelles de *firewall*)
- Linux route entre différentes interfaces :
  - physiques (par exemple `eth0`)
  - *tap* : point d'accès à une machine virtuelle KVM ou Xen
    (dans Xen, elles s'appellent *vif*, mais c'est la même chose)
  - *veth* : similaire à *tap*, point d'accès à un conteneur LXC.
    C'est en fait une paire d'interfaces, l'une étant dans l'hôte, l'autre dans le conteneur.
  - *tun* : similaire à *tap*, point d'accès à un VPN L3 (cf OpenVPN)
  - *bridge* : permet d'agréger des interfaces dans le même réseau L2
  - ...

Cas 1 – VM routée vers l'extérieur



- Deux réseaux L2 (1.0.0.0/8 et 2.0.0.0/8)
- Host1 et Host2 peuvent communiquer, Host1 et VM1 aussi
- Mais pas VM1 et Host2
- Il faut faire du routage sur Host1. Ajouter une route par défaut sur VM1, puis deux solutions :
  - A : sur Host2, une route : pour 2.0.0.0/8, la *gateway* est Host1
  - B : Faire du NAT sur Host1 pour cacher VM1

Cas 1 – Topologie physique équivalente



Cas 1 – Mise en œuvre

- Note : dans les slides, les plages d'adresses 1.0.0.0/8 et 2.0.0.0/8 sont utilisées, pour que les schémas restent lisibles. Dans la réalité, il faudrait probablement utiliser des plages d'adresses dans les plages réservées à un usage local (RFC1918) : 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- Activer le routage sur Host1 :
  ```
  Host1# echo 1 > /proc/sys/net/ipv4/ip_forward
  ```
- Indiquer à VM1 comment atteindre le reste du monde :
  ```
  VM1# ip route add default via 2.0.0.2
  ```
- Deux solutions :
  - Route sur Host2 pour lui indiquer comment atteindre VM1 :
    ```
    Host2# ip route add 2.0.0.0/8 via 1.0.0.1
    ```
  - NAT sur Host1 pour *cacher* VM1 :
    ```
    Host1# iptables -t nat -A POSTROUTING -o eth0 -j
    MASQUERADE
    ```
  - Question : quels sont les avantages de chaque solution ?

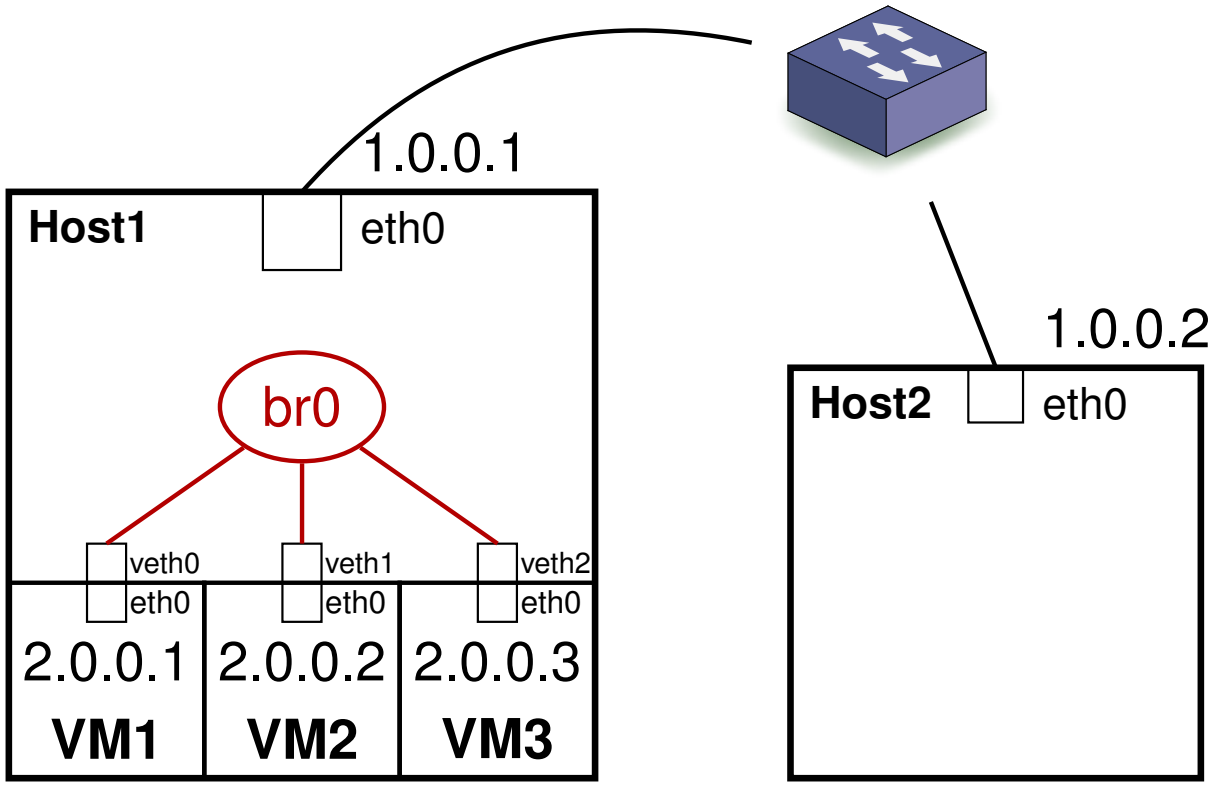
Rappel : Network Address Translation (NAT)

- Une seule IP visible de l'extérieur cache plusieurs IP privées
- Réécriture de l'en-tête IP (adresse et port source) par le routeur
- Table de correspondance renseignée et utilisée au retour
- Lorsqu'un paquet est envoyé de VM1 vers Host2 :
  - l'adresse source est celle de VM1 jusqu'à Host1
  - Host1 réécrit l'en-tête et met sa propre adresse comme source
  - Host2 reçoit le paquet provenant de Host1, et répond à Host1
  - Lorsque la réponse arrive à Host1, celui-ci réécrit l'en-tête grâce à une table de correspondance pour remettre l'IP de VM1
- Voir la table de correspondance : `conntrack -L`

Utilisation du *bridge* Linux

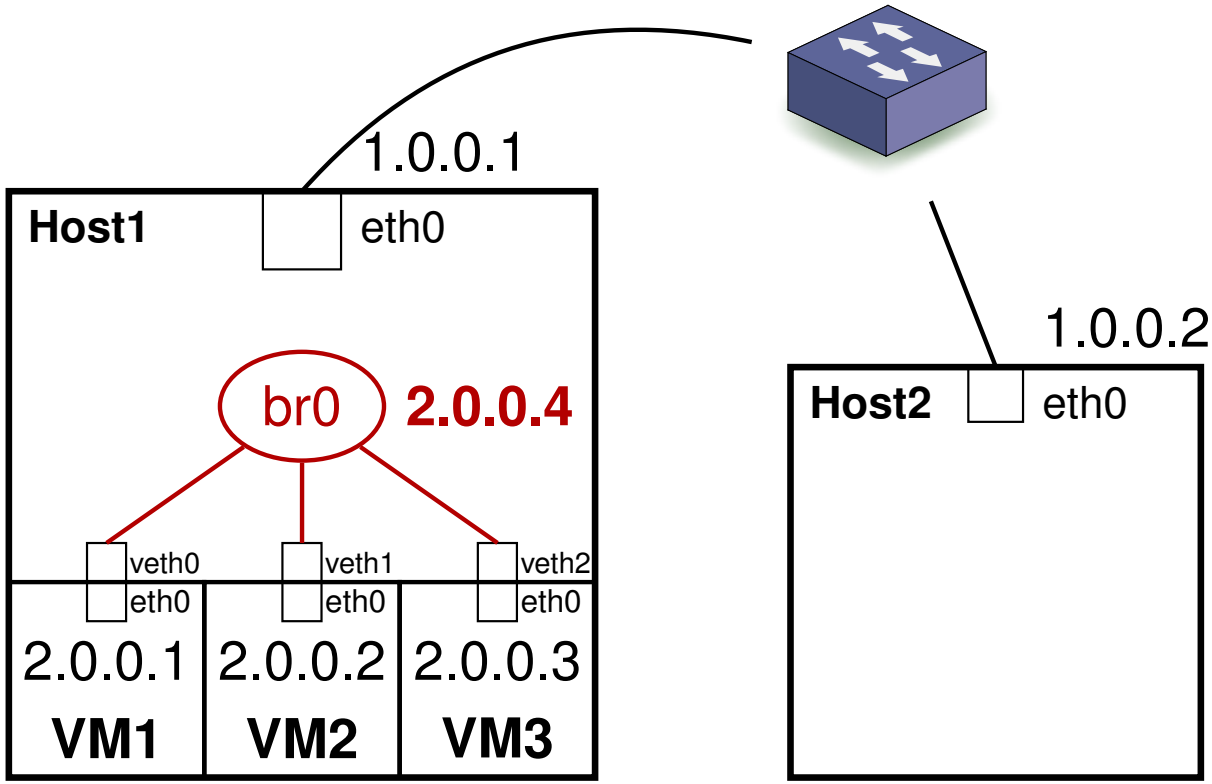
- Équivalent à un *switch* réseau – câblage virtuel
- Permet de rassembler plusieurs machines dans le même réseau L2
- Utilisation : (installer le paquet `bridge-utils`)
  - Créer un *bridge* br0 :
    `brctl addbr br0`
  - Ajouter une interface eth0 à br0 :
    `brctl addif br0 eth0`
  - Visualiser la liste des interfaces :
    `brctl show br0`
  - Visualiser la table de commutation du *switch* :
    `brctl showmacs br0`
  - On peut affecter une adresse IP au *bridge* pour accéder au réseau L2 qu'il interconnecte (mais ce n'est pas obligatoire)

Cas 2.1 – réseau sans accès extérieur



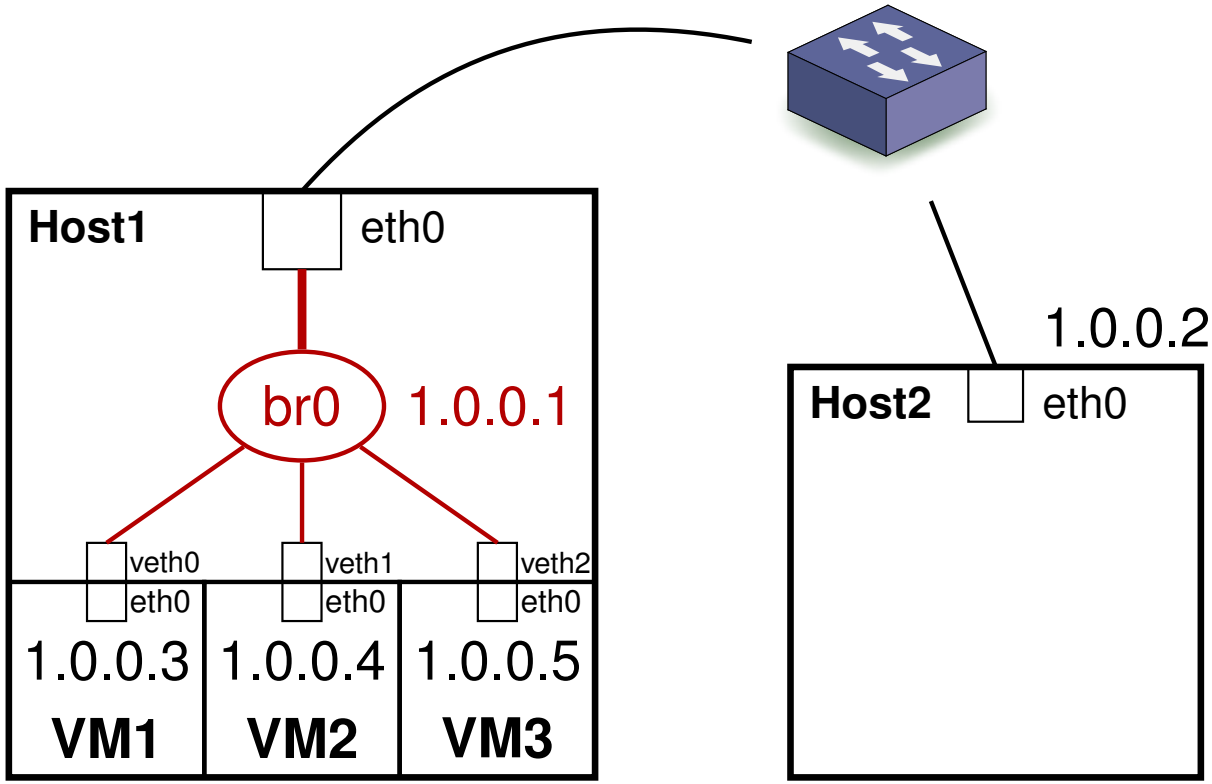
- VM1, VM2, VM3 sont sur le même réseau L2, et peuvent communiquer
- Mais elles sont isolées, et ne peuvent communiquer qu'entre elles

Cas 2.2 – *bridge* avec IP



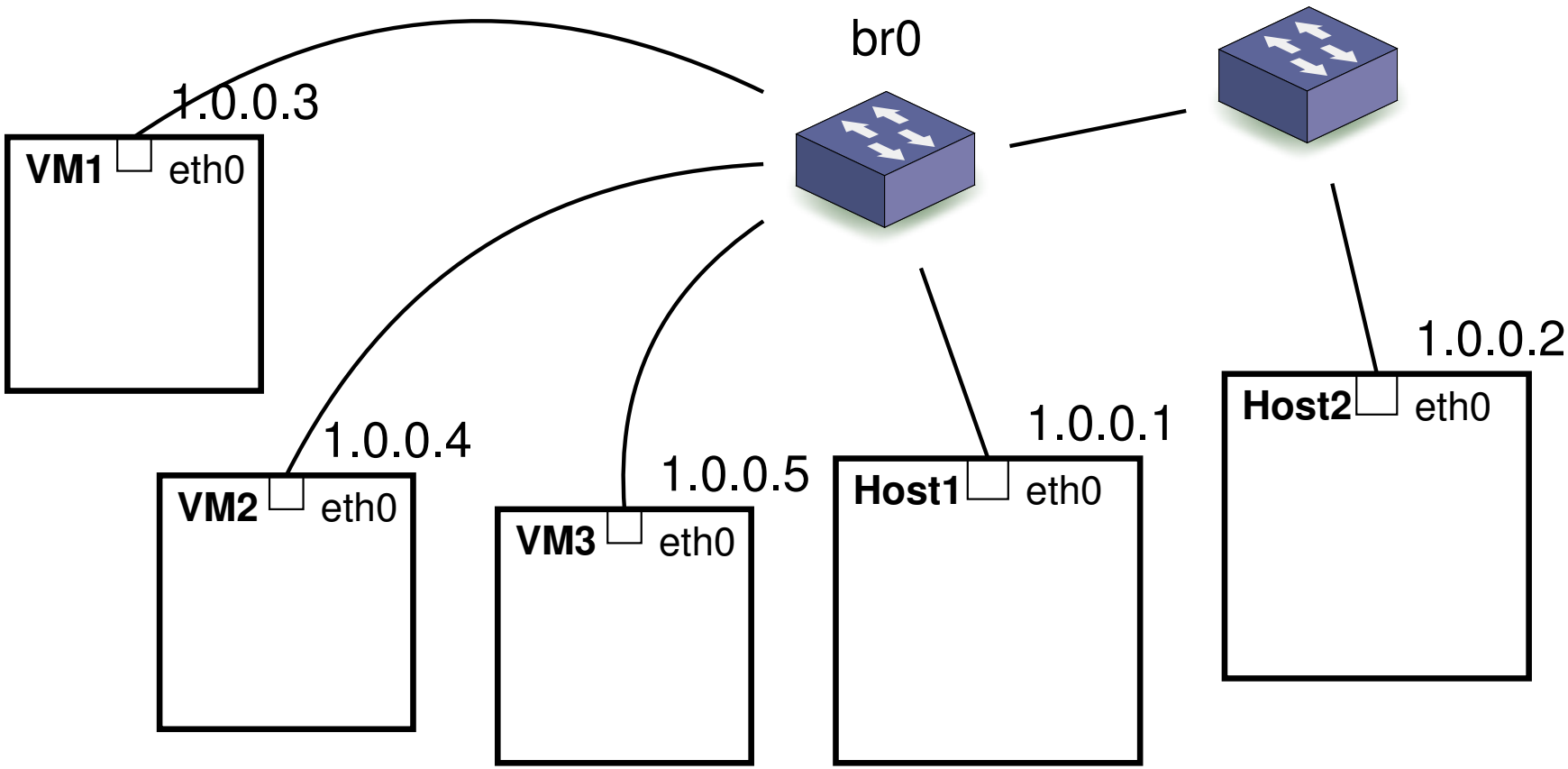
- VM1, VM2, VM3 sont sur le même réseau L2, et peuvent communiquer
- Elles peuvent aussi communiquer avec Host1, via l'IP 2.0.0.4
- Pour qu'elles puissent communiquer avec Host2, il faudrait utiliser du routage ou du NAT (cf cas 1)

Cas 2.3 – eth0 dans le *bridge*



- Host1, Host2, VM1, VM2, VM3 sont sur le même réseau L2, et peuvent communiquer directement, sans routage
- L'adresse IP pour joindre Host1 est celle du *bridge* (eth0 n'a plus d'adresse IP)

Cas 2.3 – Topologie physique équivalente

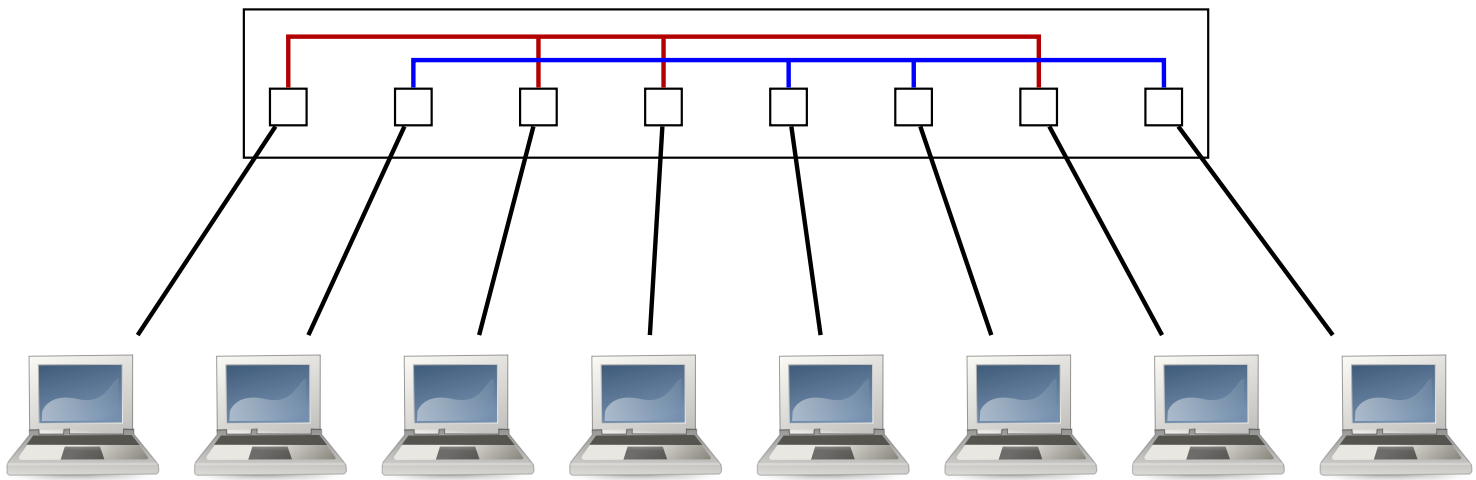


Aller plus loin

- Tout ce qui a été vu précédemment est généralisable lorsqu'il y a plusieurs machines hôtes
- On peut aussi mettre plusieurs interfaces dans chaque machine virtuelle, et mettre un bridge dans une machine virtuelle
- Il est possible de combiner cela avec l'utilisation de VLANs
- Autre alternative : utiliser un réseau *overlay* (VPN multi-point) entre les machines virtuelles situées sur différentes machines hôtes, avec VXLAN
  - Utilisé dans OpenStack Neutron pour permettre à chaque utilisateur de disposer de son propre réseau virtuel

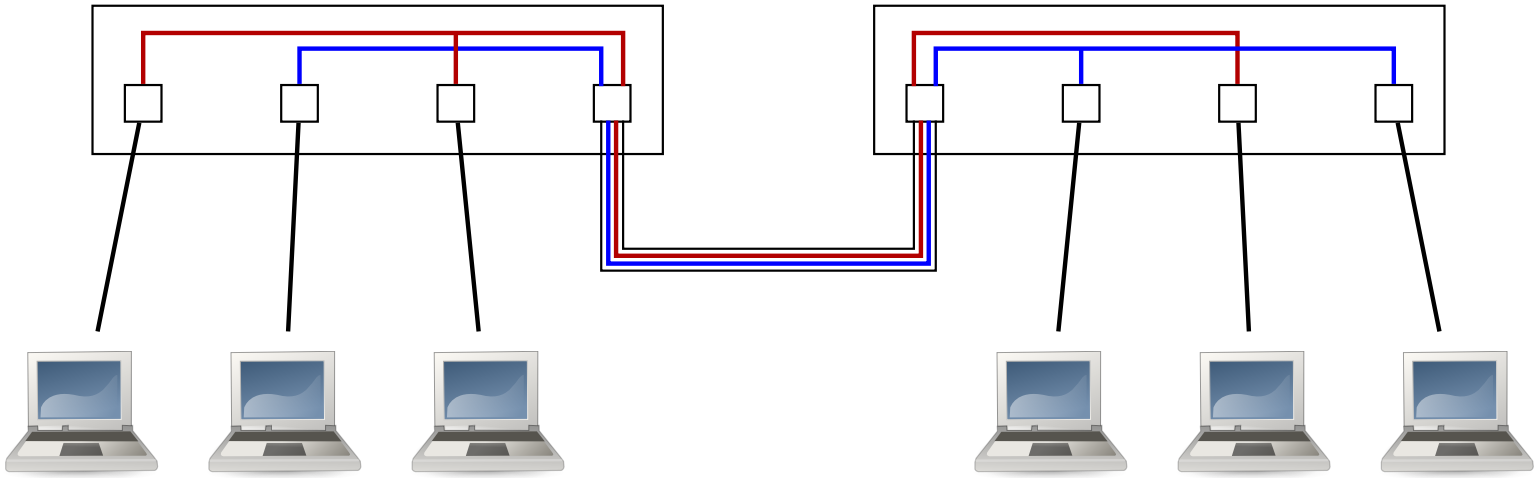
VLAN (802.1q)

- Segmenter un réseau L2 en plusieurs réseaux L2
- Séparer des usages différents, assurer une meilleure étanchéité, etc.
- Configuration par port, sur un switch :

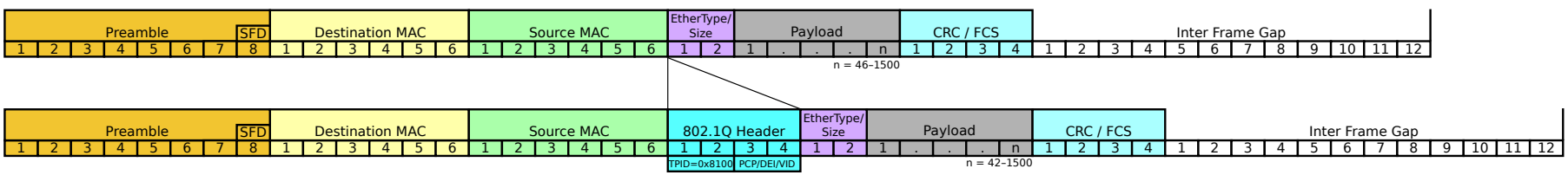


VLAN (802.1q) : ports et liens *trunk*

- Pour propager plusieurs VLAN sur un lien entre deux équipements



- Les trames ethernet sont *taggées* avec le numéro de VLAN



- Sur un lien *trunk*, on peut aussi mélanger trames *taggées* et *non taggées* (dans le VLAN par défaut)

VLAN (802.1q) : ports et liens *trunk* avec Linux

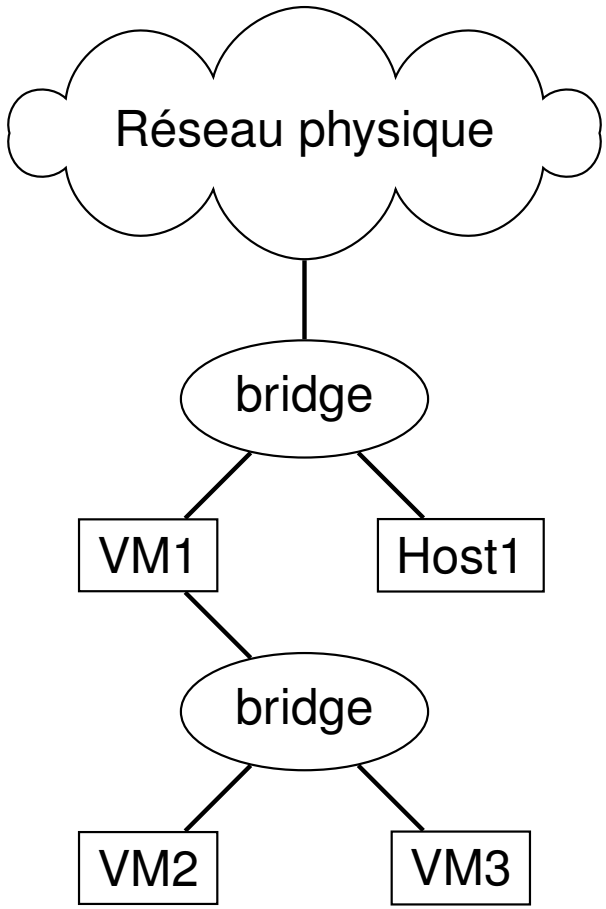
- On peut propager un lien *trunk* jusqu'à une machine Linux
- Ajouter une interface VLAN à une interface physique :
  `ip link add link eth0 name eth0.100 type vlan id 100`
- Visualiser la configuration VLAN des interfaces :
  `ip -d link show`
- Supprimer une interface VLAN :
  `ip link delete eth0.100`
- On peut affecter des adresses IPs aux interfaces VLAN, ou les mettre dans un *bridge*
- Les paquets transmis sur l'interface sans VLAN (`eth0`) sont *non taggés*

Utilisation des *network namespaces*

- *network namespaces* (*netns*) = isolation réseau dans LXC
- Voir `ip netns help`
- Créer un nouvel environnement virtuel :
  `ip netns add vm1`
- Créer une paire d'interfaces *veth* (ext0/int0) :
  `ip link add ext0 type veth peer name int0`
- Déplacer int0 dans vm1 :
  `ip link set dev int0 netns vm1`
- Renommer l'interface int0 en eth0 (pour utiliser les noms des schémas) :
  `ip link set dev int0 name eth0` (à faire dans le netns)
- Obtenir un shell dans vm1 :
  `ip netns exec vm1 bash`
- Pour savoir si vous êtes à l'intérieur du *netns* ou pas, regardez les interfaces avec `ip l` ou `ip a`

Travaux pratiques

1. Écrire des scripts en utilisant des *netns*, pour mettre en place les infrastructures décrites dans les cas 1, 2.1, 2.2, 2.3
2. Créer la topologie virtuelle ci-dessous, et vérifier son bon fonctionnement
   Il est conseillé de faire du NAT sur VM1



Travaux pratiques (2) – VLAN

- Dans l'hôte, créez un bridge, ainsi que 4 machines virtuelles, avec un couple d'interfaces *veth* pour chaque machine virtuelle. Reliez les interfaces *veth* au bridge
- Après les avoir configurées correctement, vérifiez que vous arrivez à communiquer entre les machines virtuelles
- Choisissez deux VM. Dans chacune, ajoutez une interface VLAN (avec le même numéro de VLAN), et vérifiez que vous arrivez à faire communiquer les deux VMs ensemble. Vérifiez qu'une troisième VM ne peut pas communiquer avec les deux premières si elle n'utilise pas le même numéro de VLAN

FAQ

Autres commandes utiles :

- `ip addr add 10.0.0.1/8 dev ext0`
- `ip link set dev ext0 up`
- Pour désactiver le NAT, il faut (1) enlever la règle iptables ; (2) vider le cache conntrack, avec `conntrack -F`

Troubleshooting :

1. Arrêter NetworkManager pour éviter les conflits
2. Vérifier que les interfaces (y compris les bridges) sont *up*
3. Vérifier le contenu des règles de *firewall* (`iptables -vL`, `iptables -t nat -vL`)
4. Utiliser `tcpdump` ou `tshark` ou `wireshark`
5. Par défaut, le *bridge* attend 15 secondes avant de commencer à *forwarder* les trames. Utiliser `brctl setfd br0 0` pour supprimer ce délai.
6. Une VM ne peut pas se pinguer elle-même $\rightsquigarrow$ il faut mettre l'interface lo à UP

Liens utiles

- Deux présentations très techniques sur le fonctionnement interne du réseau sous Linux :
  - `https://www.slideshare.net/ThomasGraf5/linuxcon-2015-linux-kernel-networking-walkthrough`
  - `https://www.slideshare.net/ThomasGraf5/linux-networking-explained`