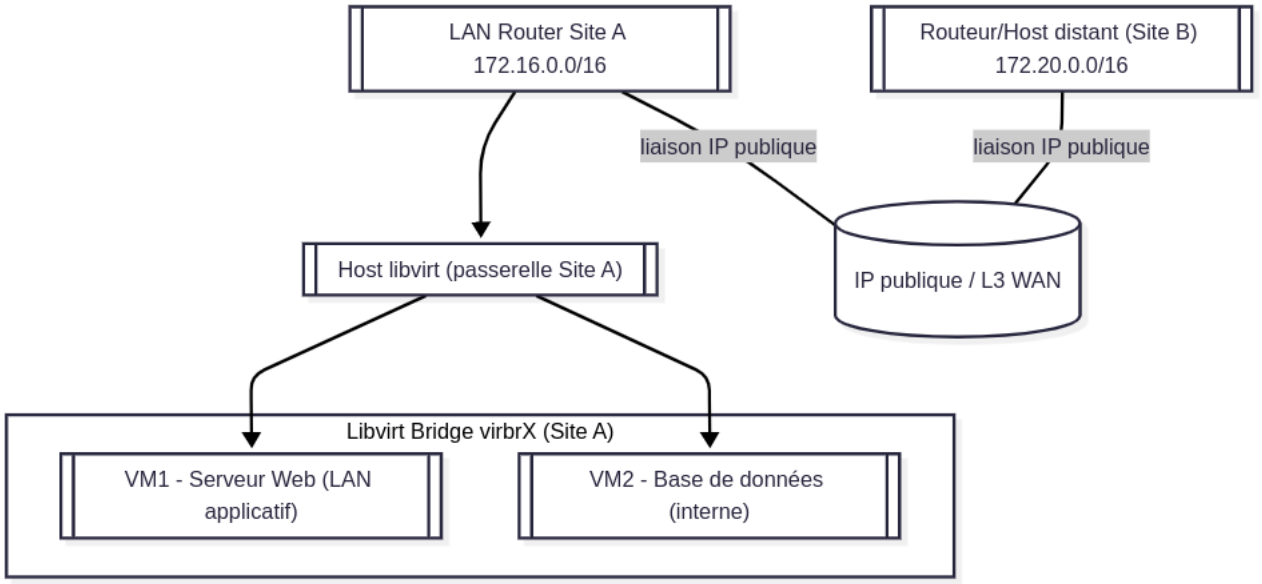
TD

2025-2026

ARES - Architecture Réseaux Entreprises

Exercice - Subnetting, routage, sécurité Libvirt + interconnexion IP-in-IP / GRE

On considère l'architecture suivante, basée sur un réseau de (notation historique 'classe B', on travaille en CIDR)



Contexte du scénario

Sites.

- **Site A** : un **LAN Router** sur 172.16.0.0/16, un **host libvirt** (passerelle) et deux VMs (**VM1** exposée au LAN ; **VM2** interne).
- **Site B** : une **seconde machine** (routeur ou host Linux) souhaitant accéder au **LAN applicatif de VM1 via un tunnel IP-in-IP ou GRE** au-dessus de la liaison IP entre R et R2.
- Les VMs sont sur un(s) sous-réseau(x) découpé(s) depuis 172.16.0.0/16 (VLSM).

1 Travail demandé

1.1 Plan d'adressage (VLSM)

Depuis **172.16.0.0/16 (Site A)**, concevez **au moins deux sous-réseaux** :

1. **Management** (Host ↔ VMs).
2. **Applicatif** (LAN ↔ VM1).

Indiquez pour chacun : adresse réseau, masque (CIDR & décimal), broadcast, plage utilisable. Attribuez des IP au **host**, **VM1**, **VM2**.

Contrainte : prévoir $\geq$ **30 hôtes** sur le LAN applicatif.

Tableau récapitulatif (à compléter).

Segment	CIDR	Masque	Réseau / Broadcast	Plage utilisable
Management				
Applicatif				

1.2 Modes Libvirt

Rappelez **NAT / Routed / Bridge** et justifiez le mode choisi ici (VM1 accessible depuis le LAN ; VM2 interne). Proposez une variante : **VM1 en Routed**, **VM2 en NAT**, impacts sur le routage.

1.3 Routage LAN (Site A)

Calculez et écrivez **les routes statiques** à configurer sur le **LAN Router (R)** pour joindre **tous les sous-réseaux VM** via l'IP du **host libvirt** (passerelle).

Syntaxe générique (exemples).

```
# Sur un Linux (routeur R)
ip route add <prefixe_VM> via <IP_host_libvirt> dev <if_lan>
```

1.4 Configuration Host libvirt

Décrire :

— IP côté LAN + configuration du pont `virbrX`.
— Activation du **forwarding IP**.
— Routes éventuelles côté host.
— Justifier l'absence ou la présence de **NAT** selon le mode choisi.

1.5 Configuration des VMs

— Fichiers réseau pour **VM1** (web, joignable depuis LAN) et **VM2** (DB, joignable depuis VM1 uniquement).
— **Passerelle** et **DNS** à définir.
— Vérifier la cohérence des routes par défaut.

2 Interconnexion Site B via IP-in-IP / GRE

2.1 Spécification du tunnel

Vous devez interconnecter **Site B** (172.20.0.0/16) au **LAN applicatif de VM1 sans exposer VM2**.

Choisissez **IP-in-IP (mode ipip) ou GRE** (justifiez le choix : encapsulation, multiprotocole, overhead).

Définir :

— Les **endpoints** du tunnel (adresses IP côté R/R2 ou H/R2 selon votre design).
— L'**adresse du lien tunnel** (ex. /30 ou /31) à affecter à `tun0/gre0`.
— Le **plan de routage** : quelles routes spécifiques doivent transiter par le tunnel ?

Architectures possibles (choisir et justifier).

A. Tunnel **R** $\leftrightarrow$ **R2** et routes vers le LAN applicatif via **H**.
B. Tunnel **H** $\leftrightarrow$ **R2** (le host libvirt termine le tunnel).

2.2 Mise en œuvre (Linux)

Option IP-in-IP.

```
# Sur l'endpoint A
ip tunnel add ipipA mode ipip remote <IP_B> local <IP_A>
ip addr add <IP_tunA>/<prefix> dev ipipA
ip link set ipipA up

# Sur l'endpoint B
ip tunnel add ipipB mode ipip remote <IP_A> local <IP_B>
ip addr add <IP_tunB>/<prefix> dev ipipB
ip link set ipipB up
```

Option GRE.

```
# Sur l'endpoint A
ip tunnel add greA mode gre remote <IP_B> local <IP_A> ttl 64
ip addr add <IP_tunA>/<prefix> dev greA
ip link set greA up

# Sur l'endpoint B
ip tunnel add greB mode gre remote <IP_A> local <IP_B> ttl 64
ip addr add <IP_tunB>/<prefix> dev greB
ip link set greB up
```

Ajoutez ensuite les **routes** pour atteindre **uniquement** le **LAN applicatif de VM1** via l'interface tunnel.

> **À calculer par vous** : le **préfixe exact** du LAN applicatif.

2.3 MTU, PMTUD, keepalive

— Calculez l'**overhead** d'encapsulation (IP-in-IP vs GRE) et **ajustez la MTU** :

```
# Rappel d'ordres de grandeur d'overhead (à justifier dans votre copie) :
# IP-in-IP : +20 octets (en-tête IPv4)
# GRE (IPv4 sur IPv4) : +24 octets (IPv4 20 + GRE 4) ; avec options possibles
# Exemple d'ajustement MTU si WAN à 1500 :
ip link set <if_tunnel> mtu 1476   # pour GRE (~1500 - 24)
ip link set <if_tunnel> mtu 1480   # pour IP-in-IP (~1500 - 20)
```

— Activez un **keepalive** simple (p. ex. sonde ICMP via `cron`/`systemd-timer`) ou utilisez `ip -o monitor` pour détecter une chute.
— Expliquez l'impact du bit **DF**/PMTUD dans votre design (fragmentation, ICMP `Frag Needed`, ajustement MTU).

2.4 Politique de sécurité inter-sites

Filtrage au niveau des endpoints.

— Autoriser encapsulation (protos **4** pour IP-in-IP ; **47** pour GRE) entre $IP_A \leftrightarrow IP_B$.
— Autoriser **ICMP** de diagnostic entre endpoints et sur le tunnel.

Segmentation. Le Site B **ne doit atteindre que VM1 (HTTP/HTTPS)**, pas VM2.

Exemples de règles `iptables`/`nftables` (à adapter).

```
# iptables (classique) - Endpoint (H ou R)
# Autoriser proto IP-in-IP (4) ou GRE (47) entre endpoints
iptables -A INPUT  -p 4  -s <IP_B> -d <IP_A> -j ACCEPT   # IP-in-IP
iptables -A INPUT  -p 47 -s <IP_B> -d <IP_A> -j ACCEPT   # GRE
iptables -A OUTPUT -p 4  -s <IP_A> -d <IP_B> -j ACCEPT
iptables -A OUTPUT -p 47 -s <IP_A> -d <IP_B> -j ACCEPT

# Autoriser ICMP (diagnostic/PMTUD)
iptables -A INPUT  -p icmp -s <IP_B> -d <IP_A> -j ACCEPT
iptables -A OUTPUT -p icmp -s <IP_A> -d <IP_B> -j ACCEPT

# Segmentation : depuis le tunnel vers VM1 seulement (HTTP/HTTPS)
iptables -A FORWARD -i <if_tunnel> -d <IP_VM1>/32 -p tcp -m multiport --dports 80,443 -j ACCEPT
iptables -A FORWARD -i <if_tunnel> -d <subnet_VM2> -j DROP
```

Bonus : GRE+IPsec. Proposez une architecture **GRE sur IPsec** (chiffrement, ports, overhead supplémentaire) *sans déploiement*. Discuter brièvement l'augmentation d'overhead et l'impact sur la MTU.
