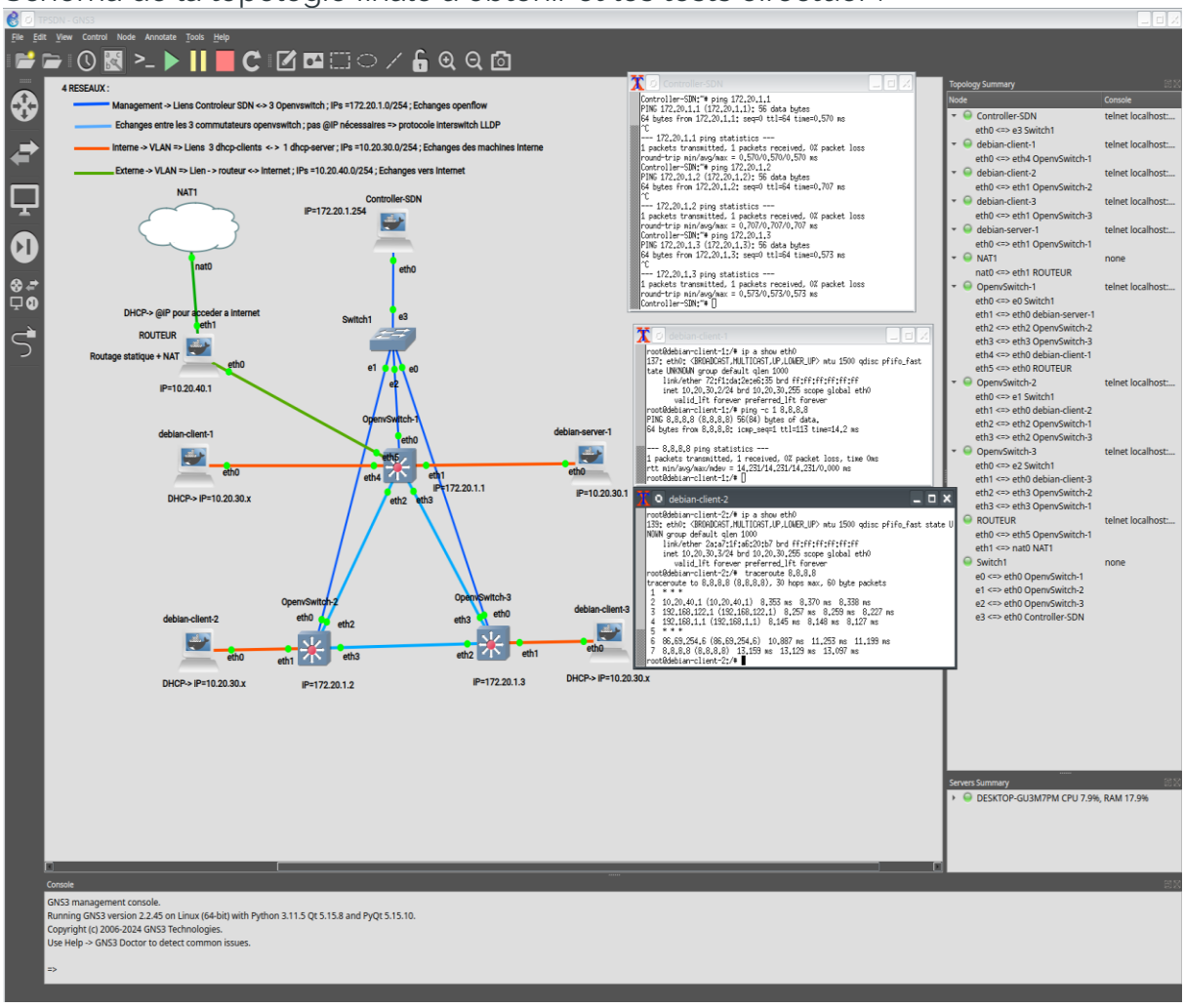
TP Openvswitch – SDN - OVN – Openflow

- Dans ce TP, l'objectif est d'utiliser l'outil GNS3 pour simuler des réseau LAN gérés par OVS+Controleur_SDN et de configurer :
  - 3 commutateurs OpenVSwitch (OVS),
  - 3 postes en DHCP pour tester les liaisons,
  - 1 serveur DHCP pour paramétrer la configuration IP des postes,
  - 1 ROUTEUR pour assurer l'interconnexion avec Internet,
  - 1 Controleur SDN Faucet (c'est un contrôleur open-source pour les réseaux OpenFlow) pour configurer les réseaux gérés.

Schéma de la topologie finale à obtenir et les tests effectuer :



L'objectif est donc de créer en 3 étapes :

1. Un réseau de management entre un contrôleur SDN et 3 commutateurs OpenVswitch (de niveau 2 et 3).
2. Un réseau (VLAN) interne entre 1 serveur dhcp et 3 clients dhcp
3. Un réseau (VLAN) externe pour accéder à internet et configurer, grâce au contrôleur SDN, les VLAN et des routeurs virtuelles pour permettre le routage entre le réseau interne et Internet.

ETAPE 1

Configurer les réseaux bleux de MANAGEMENT

- Installer un Swich simple “non OVS” : Commutateur-de-gestion
- Installer 3 conteneurs Docker OpenVSwitch
- Installer 1 conteneur Docker Faucet
- Interconnecter ces 5 équipements avec le LAN MANAGE

Configurer l’@IP des 3 équipements openvswitch précisées dans schéma précedent (@IP=172.20.1.X) en "Modifiant" les "Configuration du réseau".

```
# Static config for eth0 - Management interface
auto eth0
iface eth0 inet static
    address 172.20.1.1
    netmask 255.255.255.0
```

Configurer l’@IP du controleur FAUCET

```
# Static config for eth0 - Faucet SDN
auto eth0
iface eth0 inet static
    address 172.20.1.254
    netmask 255.255.255.0
    # Faucet needs a dummy DNS nameserver,
    # otherwise it will fail to start
    up echo nameserver 0.0.0.0 > /etc/resolv.conf
```

Lancer les 4 équipements et assurez-vous sur la console des commutateurs qu’ils ont des adresses IP de gestion comme suit :

- OpenvSwitch-1 :172.20.1.1
- OpenvSwitch-2 :172.20.1.2
- OpenvSwitch-3 :172.20.1.3
- Controleur-SDN: 172.20.1.254

Vérifier les interconnexions depuis la console du controleur-SDN : ping 172.20.1.X …

Configurer sur la console de chaque pont OVS l’IP du Controleur SDN FAUCET :

ATTENTION : Chaque pont doit avoir un id différent et il faut préciser le port qui mène au contrôleur .

```
$ ovs-vsctl set bridge br0 other-config:datapath-id=0000000000000001 -- set bridge br0
fail_mode=secure -- set-controller br0 tcp:172.20.1.254:6653 -- set controller br0 connection-
mode=out-of-band -- --if-exists del-port eth0
```

Verifier sur la console de chaque pont OVS :

```
$ ovs-vsctl get-controller br0
# => tcp:172.20.1.254:6653
```

ETAPE 2

Configurer le réseau orange Interne

- Installer 1 conteneur Docker dhcp-server
- Installer 3 conteneurs Docker dhcp-client
- Interconnecter ces 4 équipements aux Swtchs OVS pour contruire le VLAN Interne
- Configurer l'@IP du serveur DHCP (@IP=10.20.20.1) en "Modifiant" sa "Configuration du réseau".
- Lancer les 4 nouveaux équipements

Remarque : Pour éviter de lancer le Controlleur SDN faucet à la main la prochaine fois, vous pouvez spécifiez la commande de démarrage qui doit être exécutée au démarrage du conteneur (menu configurer) : `sh -c 'cd; faucet & sleep 10; ash -i -l'`

Depuis la console, configurer le Controleur SDN FAUCET (fichier : /etc/faucet/faucet.yaml.

Ce fichier permet de configurer les switchs OVS:

- les connexions avec les switchs OVS de niveau 2 (VLANs Interne et Externe)
- Les connexions de niveau 3 (routage + routeur virtuel qui devra généré et configuré)

```
$ cp /etc/faucet/faucet.yaml /etc/faucet/faucet.yaml.orig
$: cat  > /etc/faucet/faucet.yaml
Controller-SDN:~# cat /etc/faucet/faucet.yaml
vlans:
   interne:
      vid: 100
      description: "vlan interne"
      faucet_mac: "00:00:00:00:00:11"
      faucet_vips: ["10.20.30.254/24"]
   externe:
      vid: 200
      description: "vlan acces internet"
      faucet_mac: "00:00:00:00:00:22"
      faucet_vips: ["10.20.40.254/24"]
      routes:
         - route:
            ip_dst: "0.0.0.0/0"
            ip_gw: "10.20.40.1"

routers:
   router-interne-externe:
      vlans: [interne, externe]
```

```
dps:
    openvswitch-1:
        dp_id: 0x1
        hardware: "Open vSwitch"
        stack:
            priority: 1
        interfaces:
            2:
                name: "eth1"
                native_vlan: interne
            5:
                name: "eth4"
                native_vlan: interne
            6:
                name: "eth5"
                native_vlan: externe
            4:
                name: "eth3"
                description: "Link openvswitch-1 - openvswitch-3"
                stack:
                    dp: openvswitch-3
                    port: 4
            3:
                name: "eth2"
                description: "Link openvswitch-1 openvswitch-2"
                stack:
                    dp: openvswitch-2
                    port: 3

    openvswitch-2:
        dp_id: 0x2
        hardware: "Open vSwitch"
        interfaces:
            2:
                name: "eth1"
                native_vlan: interne
            3:
                name: "eth2"
                description: "Link openvswitch-2 - openvswitch-1"
                stack:
                    dp: openvswitch-1
                    port: 3
            4:
                name: "eth3"
                description: "Link openvswitch-2 - openvswitch-3"
                stack:
                    dp: openvswitch-3
                    port: 3
```

```
    openvswitch-3:
      dp_id: 0x3
      hardware: "Open vSwitch"
      interfaces:
        2:
          name: "eth1"
          native_vlan: interne
        3:
          name: "eth2"
          description: "Link openvswitch-3 - openvswitch-2"
          stack:
            dp: openvswitch-2
            port: 4
        4:
          name: "eth3"
          description: "Link openvswitch-3 - openvswitch-1"
          stack:
            dp: openvswitch-1
            port: 4  ^D
```

Testez que votre fichier de configuration est bon. Le succès affichera une sortie au format JSON de votre configuration. L'échec affichera un message d'erreur :

```
$ check_faucet_config /etc/faucet/faucet.yaml
```

Déclenchez le controleur SDN pour recharger et utiliser la nouvelle configuration.

```
$ pkill -HUP ryu-manager     ou    arréter er redémarrer le conteneur
```

Configurer le serveur DHCP

```
# cp /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf .orig

# cat > /etc/dhcp/dhcpd.conf

subnet 10.20.30.0 netmask 255.255.255.0 {
  # specifie un domaine specifique au sous-reseau
  option domain-name "mondomaine.org";
  option broadcast-address 10.20.30.255;   # adresse de diffusion
  range 10.20.30.2 10.20.30.253; # plage adresses IP dynamiques
  option routers 10.20.30.254;  # routeur par defaut
  default-lease-time 3600;
 max-lease-time 7200;
 }
```

Lancer le serveur dhcpd : `#/etc/init.d/isc-dhcp-server start`

Remarque : Pour éviter de lancer le serveur à la main la prochaine fois, vous pouvez spécifiez la commande de démarrage qui doit être exécutée au

démarrage du conteneur (menu configurer) : /gns3/bin/busybox sh -c 'export TERM=vt100;cd;service isc-dhcp-server start;while true; do /gns3/bin/busybox sh;done '

Après avoir relancer les conteneurs dhcp-client-x et depuis leurs consoles, vérifier les @IP obtenues par DHCP :

- Dhcp-clientx : 10.20.30.x

ETAPE 3

Configurer le réseau vert Externe

- Installer 1 conteneur Docker dhcp-server avec 2 interfaces nommé ROUTEUR
- Installer un accès NAT
- Interconnecter ces 2 équipements au Swtch OVS1 pour contruire le VLAN Externe
- Interconnecter le deuxième port du ROUTEUR au réseau NAT
- Configurer le port du ROUTEUR sur le réseau NAT en DHCP et l'autre port avec une @IP 10.20.40.1
    - auto eth0
    - iface eth0 inet static
    - address 10.20.40.1
    - netmask 255.255.255.0
    - broadcast 10.20.40.255

    - auto eth1
    - iface eth1 inet dhcp
    - hostname routeur

Dans cette configuration, nous allons nous intéresser au module netfilter et à iptables qui est une interface permettant de paramétrer ce module.
On rappelle que netfilter est un module du noyau Linux qui fournit :

- des fonctions de pare-feu et notamment le contrôle des machines qui peuvent se connecter, sur quels ports, de l'extérieur vers l'intérieur, ou de l'intérieur vers l'extérieur du réseau ;
- de traduction d'adresse (NAT) pour partager une connexion internet (masquerading), masquer des machines du réseau local, ou rediriger des connexions ;
- et d'historisation du trafic réseau.

Netfilter intercepte les paquets réseau à différents endroits du système (à la réception, avant de les transmettre aux processus, avant des les envoyer à la carte réseau, etc.). Les paquets interceptés passent à travers des chaînes qui vont déterminer ce que le système doit faire avec le paquet. En modifiant ces chaînes on va pouvoir bloquer certains paquets et en laisser passer d'autres. Chaque chaîne

possède des tables associées. La plus courante est la table « filter » qui permet de faire le filtrage des paquets.
Les trois chaînes principales :

- une chaîne INPUT pour filtrer les paquets à destination du système,
- une chaîne OUTPUT pour filtrer les paquets émis par les processus du système,
- et une chaîne FORWARD pour filtrer les paquets que le système doit transmettre (par ex. lorsqu'on fait du NAT).

En ajoutant des règles dans ces chaînes on pourra laisser passer ou jeter les paquets suivant certains
critères. Il est possible de créer par ailleurs ses propres chaînes.
La différence entre Netfilter et iptables est la suivante :

- Netfilter est un module, et fonctionne en mode Noyau. C'est lui qui intercepte et manipule les paquets IP avant et après le routage.
- iptables est la commande qui permet de configurer Netfilter en espace Utilisateur.

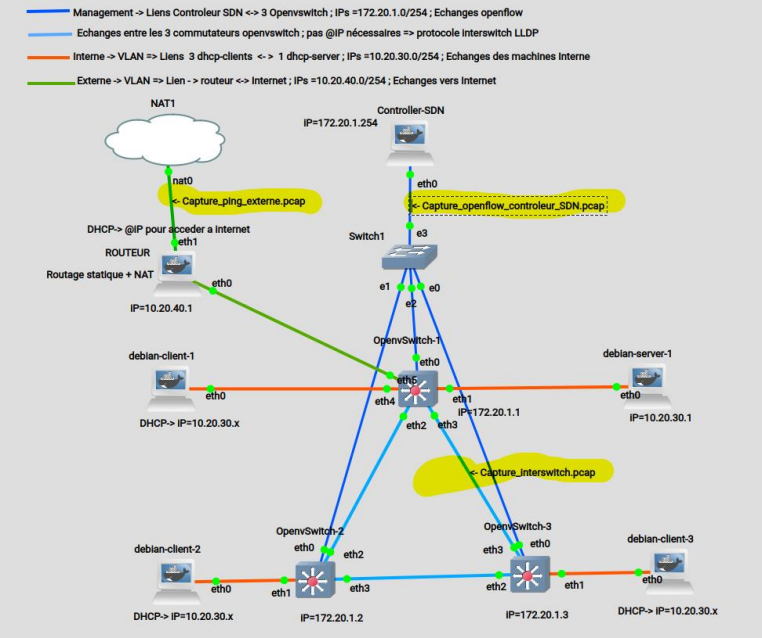
Ajoutez une régle routage pour router les trames revenant au réseau interne et une règle dans la table nat de sorte à mettre en place du SNAT dynamique (*masquerading*) permettant de remplacer l'adresse IP source privée par l'adresse IP externe associée à l'interface eth0 du nœud ROUTEUR pour tout paquet sortant via l'interface eth1.

- ip route add 10.20.30.0/24 via 10.20.40.254 ;
- iptables -v -t nat -A POSTROUTING -s 10.0.0.0/8 -o eth1 -j MASQUERADE;
- iptables -v -A FORWARD -i eth1 -o eth0 -d 10.0.0.0/8 -m state --state RELATED,ESTABLISHED -j ACCEPT

Remarque : Pour éviter de lancer le serveur à la main la prochaine fois, vous pouvez spécifiez la commande de démarrage qui doit être exécutée au démarrage du conteneur (menu configurer) : /gns3/bin/busybox sh -c 'export TERM=vt100;cd;ip route add 10.20.30.0/24 via 10.20.40.254 ; iptables -v -t nat -A POSTROUTING -s 10.0.0.0/8 -o eth1 -j MASQUERADE; iptables -v -A FORWARD -i eth1 -o eth0 -d 10.0.0.0/8 -m state --state RELATED,ESTABLISHED -j ACCEPT;while true; do /gns3/bin/busybox sh;done

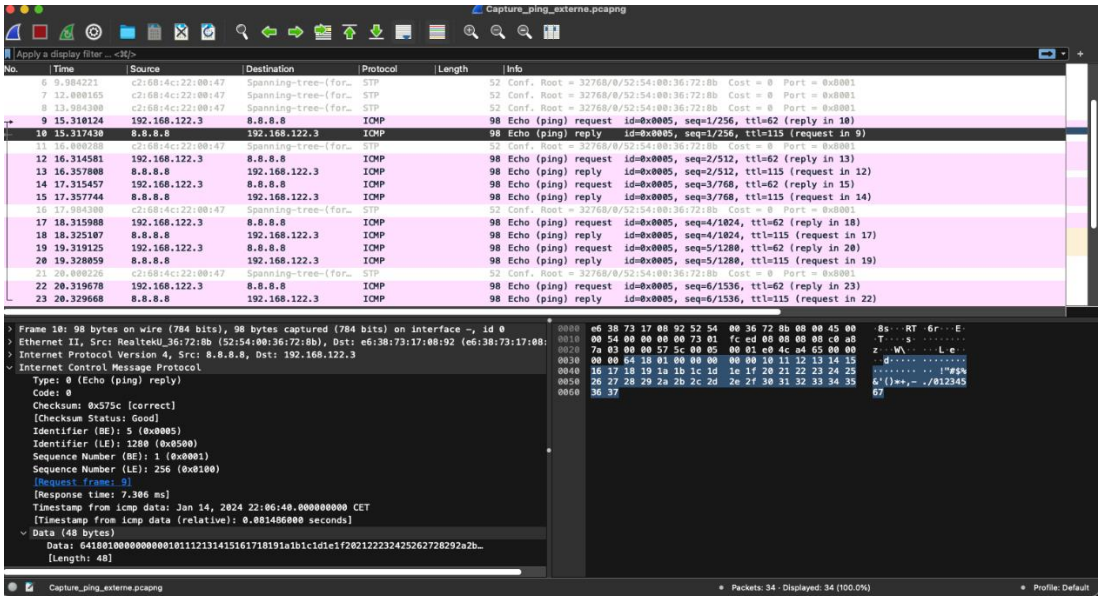
Après avoir lancé tous les équipements : ROUTEUR, Controleur-SDN, 3 dhcp-clients, 1 dhcp-server, 3 openvswitchs; Lancer un traceroute 8.8.8.8 depuis la console d'un poste

dhcp-client. J'ai installé 3 sondes wireshark pour récupérer les trames.

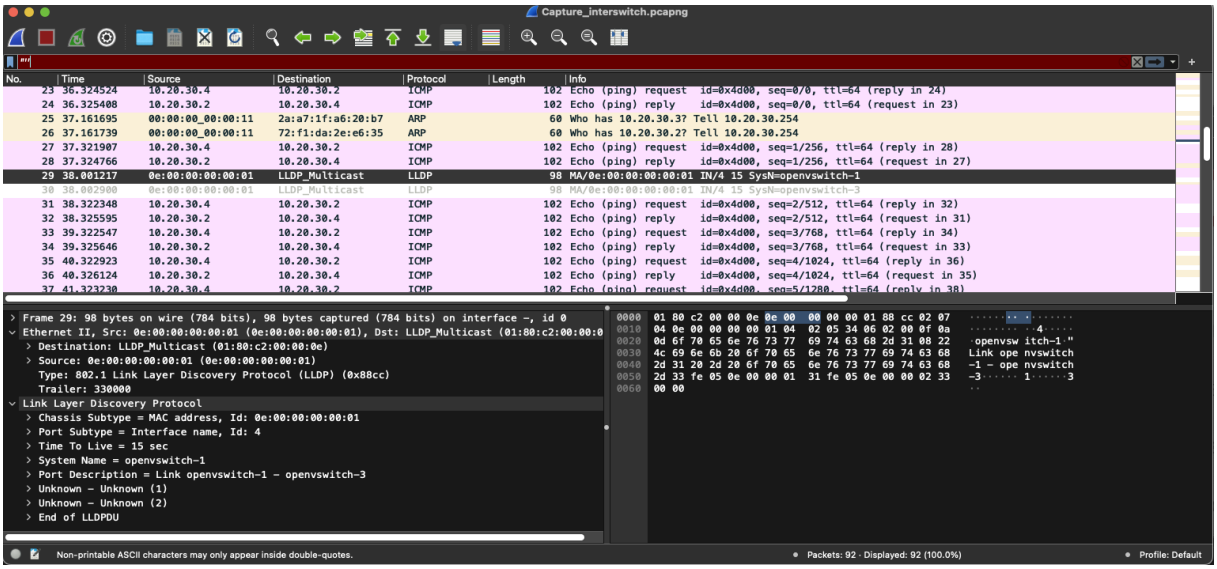


Voici les trames que vous devriez obtenir pendant ce ping 8.8.8.8 :

1- ***Trames ICMP ECHO sur le lien vers Internet***



2- ***Trames LLDP +ECHO sur les liens entre les commutateursOpenVswitchs***



3-Trames OpenFlow sur le lien du Contrôleur-SDN