

TP SDN

Diani Badr
5ISS
Master REOC: 2024/2025

Table des matières

Introduction	2
Partie 1 : Prise en main de Floodlight et de Mininet / Containernet	3
A) Démarrage du contrôleur SDN.....	3
B) Émulation d'un réseau SDN	3
C) curl	5
D) Interface Web du contrôleur Floodlight.....	6
Partie 2 : Redirection transparente de paquets IP	10
A) Personnalisation de topologie et lancement de serveurs Web.....	11
B) Redirection transparente de paquets IP.....	12
Partie 3 : Intégration de modules à Floodlight.....	15
Conclusion	17

Introduction

Ce rapport présente les travaux réalisés autour des réseaux définis par logiciel (SDN) à travers l'utilisation des outils Floodlight, Mininet et Containernet. Le but principal de ce projet est de comprendre et de mettre en œuvre les concepts fondamentaux du SDN, notamment la gestion des contrôleurs, l'émulation de réseaux, la configuration des règles de flux, et l'intégration de modules personnalisés.

Les technologies SDN, qui permettent de centraliser la gestion des réseaux, offrent des solutions flexibles et dynamiques pour répondre aux besoins croissants en matière de connectivité et de sécurité. Ce projet s'inscrit dans ce contexte et vise à renforcer la maîtrise de ces outils et concepts tout en explorant leurs applications pratiques.

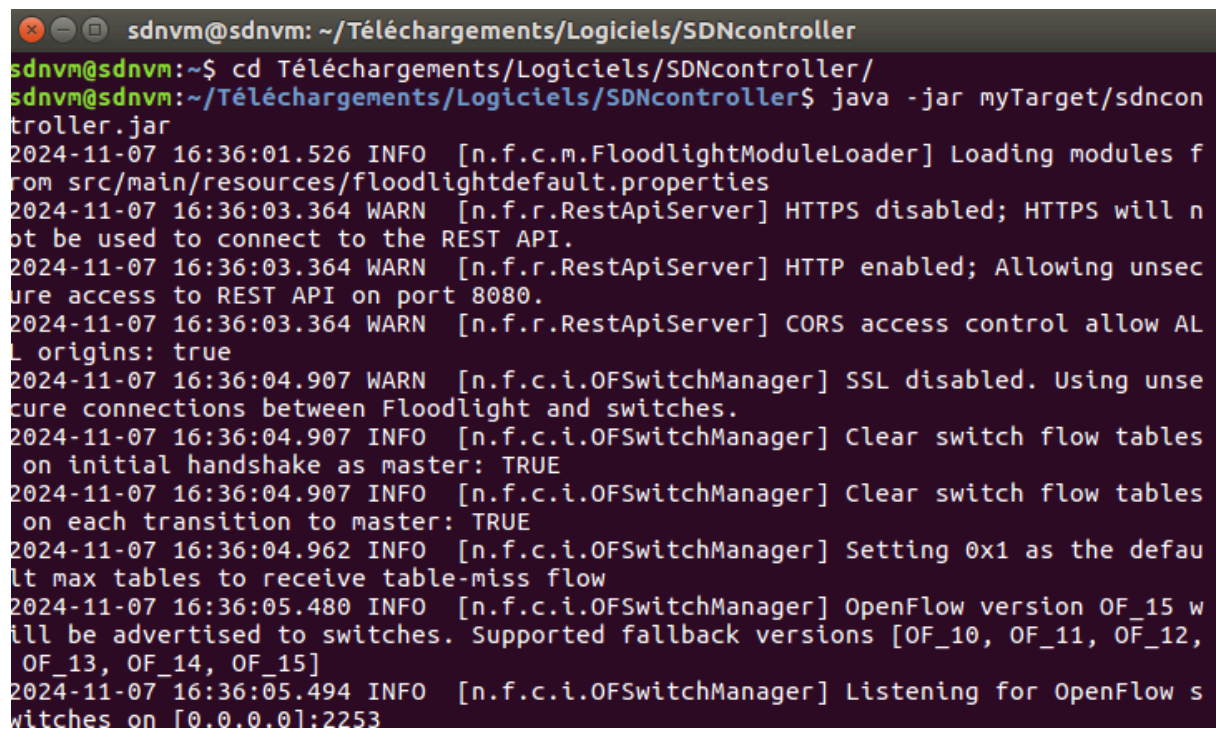
Ce rapport est structuré en plusieurs parties, détaillant les différentes étapes du projet, depuis la prise en main des outils jusqu'à l'intégration de modules spécifiques, en passant par la configuration avancée de réseaux et la redirection transparente de paquets.

Partie 1 : Prise en main de Floodlight et de Mininet / Containernet

Dans cette partie, nous traiterons du démarrage d'un contrôleur SDN depuis la ligne de commande, de l'émulation d'un réseau SDN à l'aide de Mininet ou Containernet, de l'installation de règles de flux, ainsi que de l'utilisation de l'interface Web du contrôleur Floodlight.

A) Démarrage du contrôleur SDN :

Pour démarrer correctement le contrôleur SDN, j'ai commencé par lancer l'outil "Terminal" pour accéder à la console. Ensuite, je me suis déplacé dans le dossier "SDNcontroller" à l'aide de la commande appropriée. Enfin, j'ai démarré le contrôleur en exécutant la commande nécessaire. La console a alors affiché des messages confirmant que les différents modules du contrôleur ont été lancés avec succès, comme le montre la photo ci-dessous.



```
sdnvm@sdnvm: ~/Téléchargements/Logiciels/SDNcontroller
sdnvm@sdnvm:~/Téléchargements/Logiciels/SDNcontroller$ cd Téléchargements/Logiciels/SDNcontroller/
sdnvm@sdnvm:~/Téléchargements/Logiciels/SDNcontroller$ java -jar myTarget/sdncontroller.jar
2024-11-07 16:36:01.526 INFO [n.f.c.m.FloodlightModuleLoader] Loading modules from src/main/resources/floodlightdefault.properties
2024-11-07 16:36:03.364 WARN [n.f.r.RestApiServer] HTTPS disabled; HTTPS will not be used to connect to the REST API.
2024-11-07 16:36:03.364 WARN [n.f.r.RestApiServer] HTTP enabled; Allowing unsecured access to REST API on port 8080.
2024-11-07 16:36:03.364 WARN [n.f.r.RestApiServer] CORS access control allow all origins: true
2024-11-07 16:36:04.907 WARN [n.f.c.i.OFSwitchManager] SSL disabled. Using unsecured connections between Floodlight and switches.
2024-11-07 16:36:04.907 INFO [n.f.c.i.OFSwitchManager] Clear switch flow tables on initial handshake as master: TRUE
2024-11-07 16:36:04.907 INFO [n.f.c.i.OFSwitchManager] Clear switch flow tables on each transition to master: TRUE
2024-11-07 16:36:04.962 INFO [n.f.c.i.OFSwitchManager] Setting 0x1 as the default max tables to receive table-miss flow
2024-11-07 16:36:05.480 INFO [n.f.c.i.OFSwitchManager] OpenFlow version OF_15 will be advertised to switches. Supported fallback versions [OF_10, OF_11, OF_12, OF_13, OF_14, OF_15]
2024-11-07 16:36:05.494 INFO [n.f.c.i.OFSwitchManager] Listening for OpenFlow switches on [0.0.0.0]:2253
```

B) Émulation d'un réseau SDN :



Figure 1

J'ai émulé un réseau SDN à l'aide de l'outil Mininet en utilisant la commande 'mn' avec les paramètres `--controller=remote,ip=10.0.2.15,port=2253` pour spécifier l'adresse IP du contrôleur, et `--topo=linear,2` pour créer une topologie linéaire constituée de deux hôtes et de deux switchs SDN. Cette configuration m'a permis de simuler un réseau SDN avec des connexions simples entre les éléments du réseau, permettant ainsi au contrôleur de gérer et d'orienter le trafic via les switchs.

```

sdnvm@sdnvm: ~
sdnvm@sdnvm:~$ sudo mn --topo=linear,2 --controller=remote,ip=10.0.2.15,port=2
253 --mac
[sudo] Mot de passe de sdnvm :
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:

```

La commande `h1 ping -c 1 h2` dans Mininet / Containernet sert à tester la connectivité entre deux hôtes (h1 et h2) dans le réseau émulé. Plus précisément, elle envoie un seul paquet ICMP (commande ping) de l'hôte 1 vers l'hôte 2 pour vérifier si l'hôte 2 répond correctement.

```

containernet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=366 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 366.567/366.567/366.567/0.000 ms
containernet> h2 ping -c 1 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=7.28 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.280/7.280/7.280/0.000 ms
containernet>

```

Après avoir exécuté la commande `h1 ping -c 1 h2`, on constate que l'adresse IP attribuée à Hôte 2 est 10.0.0.2. Ensuite, en inversant les rôles des hôtes avec la commande `h2 ping -c 1 h1`, on confirme que l'adresse IP de Hôte 1 est 10.0.0.1. On peut ainsi vérifier que les deux hôtes sont correctement configurés et connectés, avec un temps de réponse relativement faible (7.28 ms entre h2 et h1).

C) *curl*

Dans le cadre de la configuration d'un réseau SDN, le contrôleur dispose d'un module nommé Static Entry Pusher. Ce module, exposant une API REST, permet d'installer, modifier ou supprimer des règles de flux sur les switchs sous contrôle. L'objectif ici est d'ajouter une règle de flux sur Switch 1 pour bloquer tous les paquets en provenance de l'Hôte 1 et à destination de l'Hôte 2, empêchant ainsi toute communication directe entre ces deux hôtes..

Étapes de configuration:

- **Identification du switch SDN**

```
sdnvm@sdnvm:~$ sudo ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
1(s1-eth1): addr:6e:84:02:a6:34:90
  config:      0
  state:      0
  current:    10GB-FD COPPER
  speed: 10000 Mbps now, 0 Mbps max
2(s1-eth2): addr:be:09:55:ef:9d:55
  config:      0
  state:      0
  current:    10GB-FD COPPER
  speed: 10000 Mbps now, 0 Mbps max
LOCAL(s1): addr:d6:e4:bf:4d:21:49
  config:      PORT_DOWN
  state:      LINK_DOWN
  speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
sdnvm@sdnvm:~$
```

Le dpid (Data Path Identifier) du Switch 1 est récupéré 1. Ce paramètre est indispensable pour identifier le switch dans la règle.

- **Création de la règle de flux**

Une règle de flux a été créée pour bloquer toute communication entre l'Hôte 1 et l'Hôte 2 dans le réseau SDN. Cette règle, configurée sur le Switch 1, spécifie que tout paquet provenant de l'adresse IP source de l'Hôte 1 (10.0.0.1) et destiné à l'adresse IP de l'Hôte 2 (10.0.0.2) sera supprimé (action : DROP). De plus, cette règle s'applique aux paquets entrant par le port 1 du switch, garantissant ainsi une isolation complète entre les deux hôtes.

```
sdnvm@sdnvm:~$ curl -X POST -d '{"switch":"1", "name":"f1", "priority":"36000", "in_port":"1", "eth_type":"0x0800", "ipv4_src":"10.0.0.1", "ipv4_dst":"10.0.0.2", "actions":"DROP"}' http://10.0.2.15:8080/wm/staticflowpusher/json
{"status" : "Entry pushed"}sdnvm@sdnvm:~$
```

- Test de la règle

Pour vérifier l'efficacité de la règle, une commande ping est effectuée depuis Hôte 1 vers Hôte 2 :

```
containernet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

containernet>
```

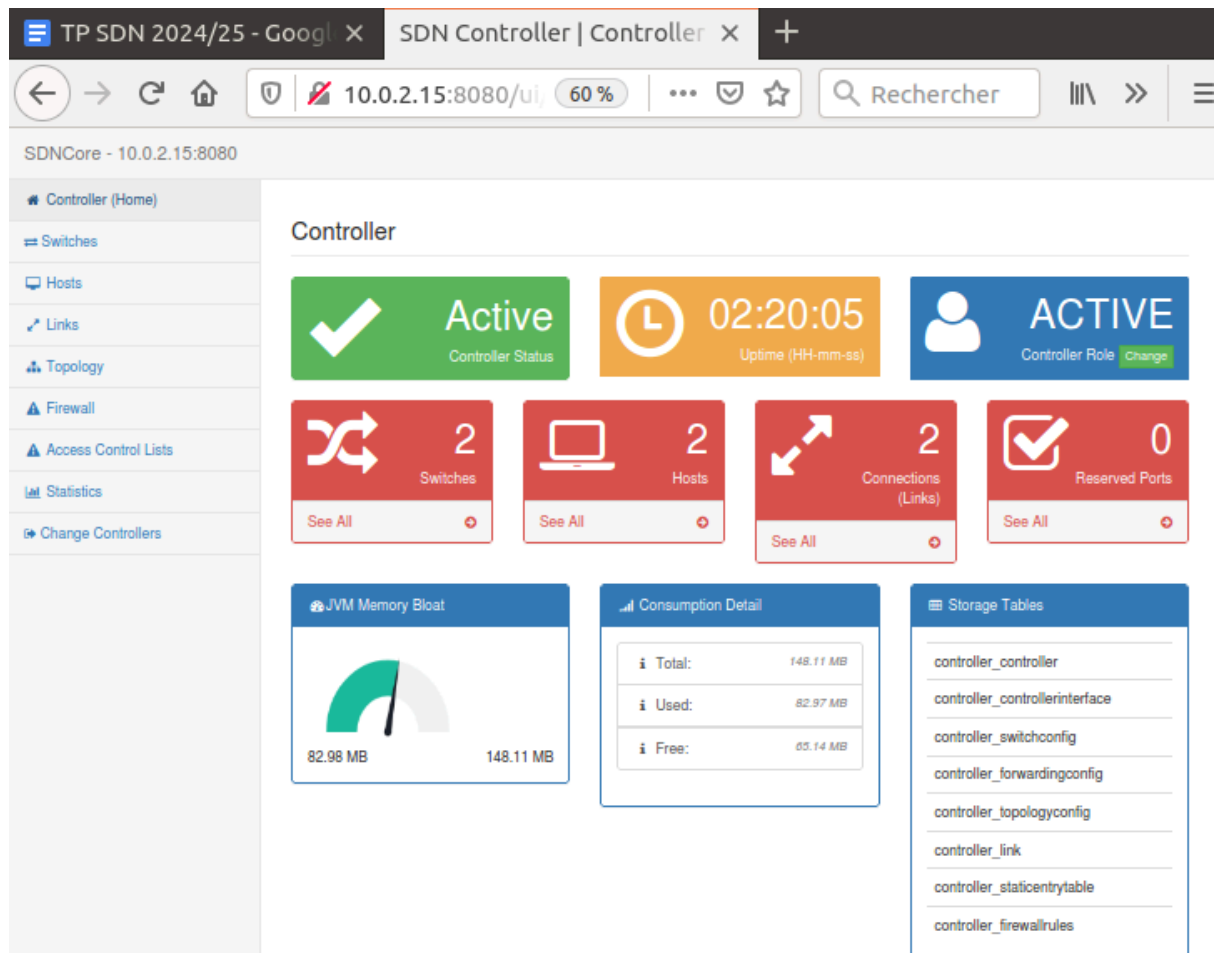
Le résultat montre une perte de 100 % des paquets, confirmant que la règle a bien été appliquée et que les paquets sont supprimés.

D) Interface Web du contrôleur Floodlight:

Le contrôleur SDN dispose d'une interface Web accessible à l'adresse 10.0.2.15:8080/ui/index.html à l'intérieur de la machine virtuelle (VM). Cette interface Web offre plusieurs fonctionnalités pour faciliter la gestion et le suivi en temps réel du réseau SDN :

1. Controller (Home)

Cette section affiche un tableau de bord général du contrôleur, comme montré dans la capture d'écran. Elle donne un aperçu rapide du statut du contrôleur (actif ou inactif), du temps de fonctionnement, du nombre de switches, hôtes et liens actifs, ainsi que l'utilisation des ressources mémoire.



2. Switches

Cette section répertorie tous les switchs SDN connectés au contrôleur. Pour chaque switch, elle fournit des détails tels que le DPID (Data Path Identifier), les ports disponibles, et les règles de flux configurées dans les tables du switch. Ces informations permettent de visualiser et de modifier les configurations des switchs pour assurer un routage et une gestion efficaces des flux réseau.

Switches

Switches Connected		
Switch ID	IPv4 Address	Connected Since
00:00:00:00:00:00:01	/10.0.2.15:43754	Tue Dec 17 2024 16:31:22 GMT+0100 (heure normale d'Europe centrale)
00:00:00:00:00:00:02	/10.0.2.15:43756	Tue Dec 17 2024 16:31:22 GMT+0100 (heure normale d'Europe centrale)
Showing 1 to 2 of 2 entries		

Switch Roles	
Switch MAC	Role
00:00:00:00:00:00:02	MASTER
00:00:00:00:00:00:01	MASTER

3. Hosts

La section "Hosts" affiche tous les hôtes connectés au réseau. Les informations comprennent l'adresse MAC, l'adresse IP, le port d'attache sur le switch, et le switch auquel chaque hôte est relié. Ces détails sont essentiels pour identifier les périphériques connectés au réseau, diagnostiquer les problèmes de connectivité, ou surveiller l'activité des hôtes.

Hosts

Hosts Connected					
MAC	IPv4 Address	IPv6 Address	Switch	Port	Last Seen
00:00:00:00:00:01	10.0.0.1	fe80::200:ff:fe00:1	00:00:00:00:00:00:01	1	1734457615440
00:00:00:00:00:02	10.0.0.2	fe80::200:ff:fe00:2	00:00:00:00:00:00:02	1	1734457615441
Showing 1 to 2 of 2 entries					

4. Links

Cette section montre les connexions physiques au sein du réseau, que ce soit entre les switchs eux-mêmes ou entre les switchs et les hôtes. Chaque lien est décrit avec des informations telles que le port source et destination. Ces données permettent d'avoir une vue claire des chemins réseau disponibles et facilitent l'identification des éventuels points de défaillance ou des goulots d'étranglement.

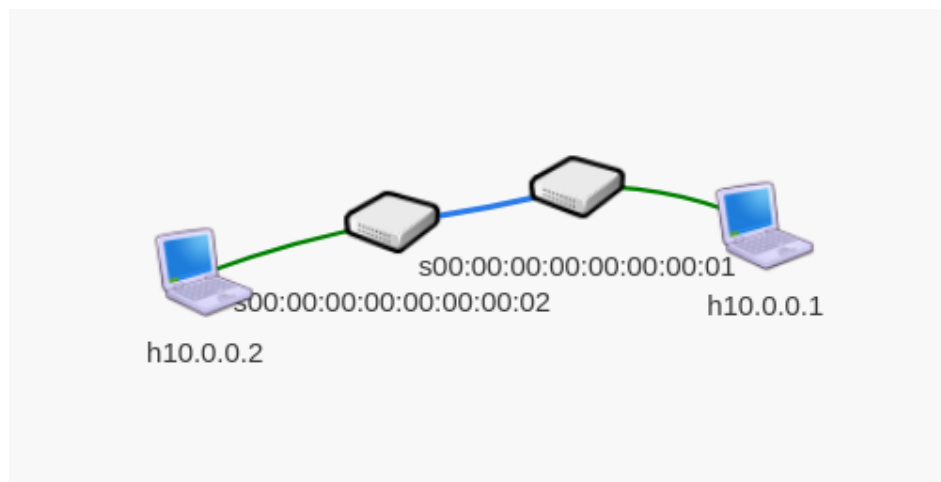
Links

Internal Links					
Direction	Source Port	Source Switch	Destination Port	Destination Switch	Type
bidirectional	2	00:00:00:00:00:00:01	2	00:00:00:00:00:00:02	internal
Showing 1 to 1 of 1 entries					

External Links					
Direction	Source Port	Source Switch	Destination Port	Destination Switch	Type
No data available in table					
Showing 0 to 0 of 0 entries					

5. Topology

La section "Topology" offre une visualisation graphique et interactive de la structure réseau. Elle affiche les switches, les hôtes et les liens, permettant une compréhension immédiate de la disposition et de la connectivité des composants. Cette vue est particulièrement utile pour les administrateurs qui souhaitent diagnostiquer des problèmes de topologie ou analyser la configuration globale du réseau.



6. Firewall

Cette section permet de configurer un pare-feu pour le réseau SDN. Les administrateurs peuvent ajouter, modifier ou supprimer des règles de sécurité pour bloquer ou autoriser certains flux réseau en fonction de critères comme l'adresse IP,

le protocole ou les ports. Le pare-feu offre un moyen efficace de sécuriser le réseau contre les menaces potentielles en limitant les accès non autorisés.

7. Access Control Lists (ACLs)

La section des ACLs permet de définir des règles de contrôle d'accès qui restreignent ou autorisent certains types de trafic en fonction de critères prédéfinis. Ces règles peuvent s'appliquer à des adresses IP spécifiques, des plages de ports, ou d'autres caractéristiques du trafic. Les ACLs fournissent un niveau supplémentaire de contrôle et de sécurité, en complément des fonctionnalités du pare-feu.

8. Statistics

La section "Statistics" fournit des données détaillées sur les performances et l'utilisation du réseau. Elle inclut des informations sur la quantité de données transmises ou reçues, le nombre de paquets traités par port ou flux, et d'autres métriques essentielles. Ces statistiques permettent de surveiller la santé du réseau, d'optimiser les performances, et de détecter les anomalies.

9. Change Controllers

Cette section permet de modifier le contrôleur actif ou de configurer des contrôleurs supplémentaires pour gérer le réseau. Cette fonctionnalité est particulièrement utile dans les configurations avancées, où plusieurs contrôleurs sont utilisés pour assurer la redondance, l'équilibrage de charge, ou la maintenance continue du réseau.

Partie 2 : Redirection transparente de paquets IP

Cette deuxième partie traite de la conception et de la mise en place de topologies réseau personnalisées dans des environnements tels que Mininet ou Containernet, permettant de simuler des scénarios variés. Elle aborde également le déploiement d'un serveur Web au sein de ces machines virtuelles, ainsi que la configuration de redirections transparentes de paquets IP, garantissant une gestion fluide du trafic réseau tout en restant imperceptible pour les utilisateurs finaux.

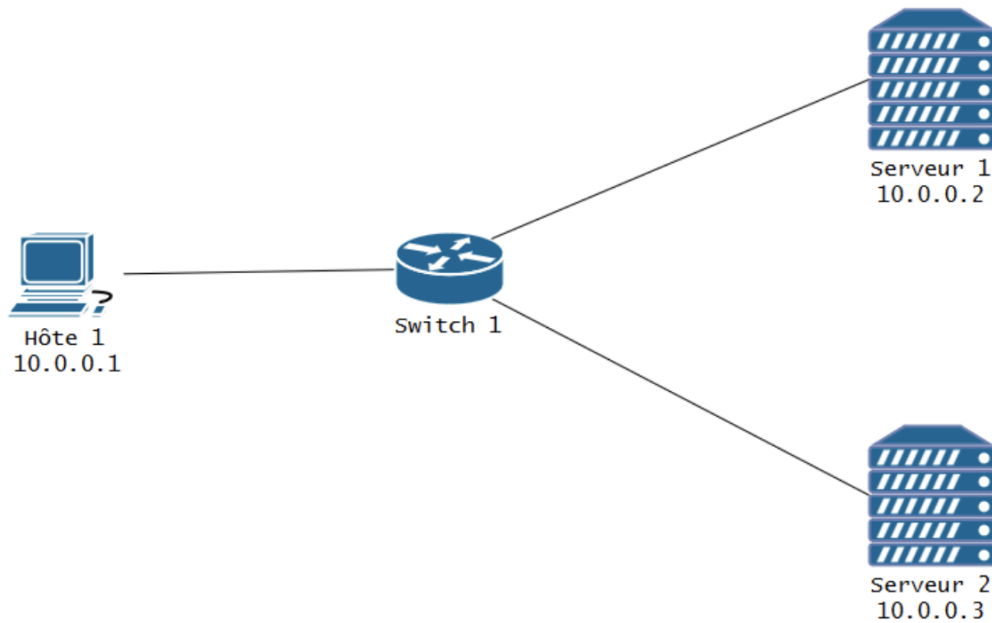


Figure 2

A) Personnalisation de topologie et lancement de serveurs Web :

Dans le cadre de ce projet, la création de la topologie réseau a été réalisée à l'aide du fichier Python topo2.py, situé dans le répertoire Topologies du dossier SDNcontroller. Ce fichier définit une topologie conforme à la figure 2, à l'exception des serveurs Web qui ne sont pas initialement lancés. La topologie a ensuite été déployée via Mininet avec les commande suivante :

```

sdnvm@sdnvm: ~/Téléchargements/Logiciels/SDNcontroller
sdnvm@sdnvm:~$ cd Téléchargements/Logiciels/SDNcontroller/
sdnvm@sdnvm:~/Téléchargements/Logiciels/SDNcontroller$ sudo mn --custom Topologies/topo2.py --topo=mytopo --controller=remote,ip=10.0.2.15,port=2253 --mac
[sudo] Mot de passe de sdnvm :
*** Creating network
*** Adding controller
*** Adding hosts:
Host1 Server1 Server2
*** Adding switches:
Switch1
*** Adding links:
(Host1, Switch1) (Switch1, Server1) (Switch1, Server2)
*** Configuring hosts
Host1 Server1 Server2
*** Starting controller
c0
*** Starting 1 switches
Switch1 ...
*** Starting CLI:
containernet>
  
```

Cette commande charge la topologie personnalisée mytopo, connecte Mininet à un contrôleur SDN distant spécifié par l'adresse IP 10.0.2.15 et le port 2253, et attribue automatiquement les adresses MAC. Une fois la topologie lancée, les serveurs Web

ont été démarrés dans l'invite de commande Mininet à l'aide des commandes suivantes :

```
containernet> Server1 python -m SimpleHTTPServer 80 &  
containernet> Server2 python -m SimpleHTTPServer 80 &
```

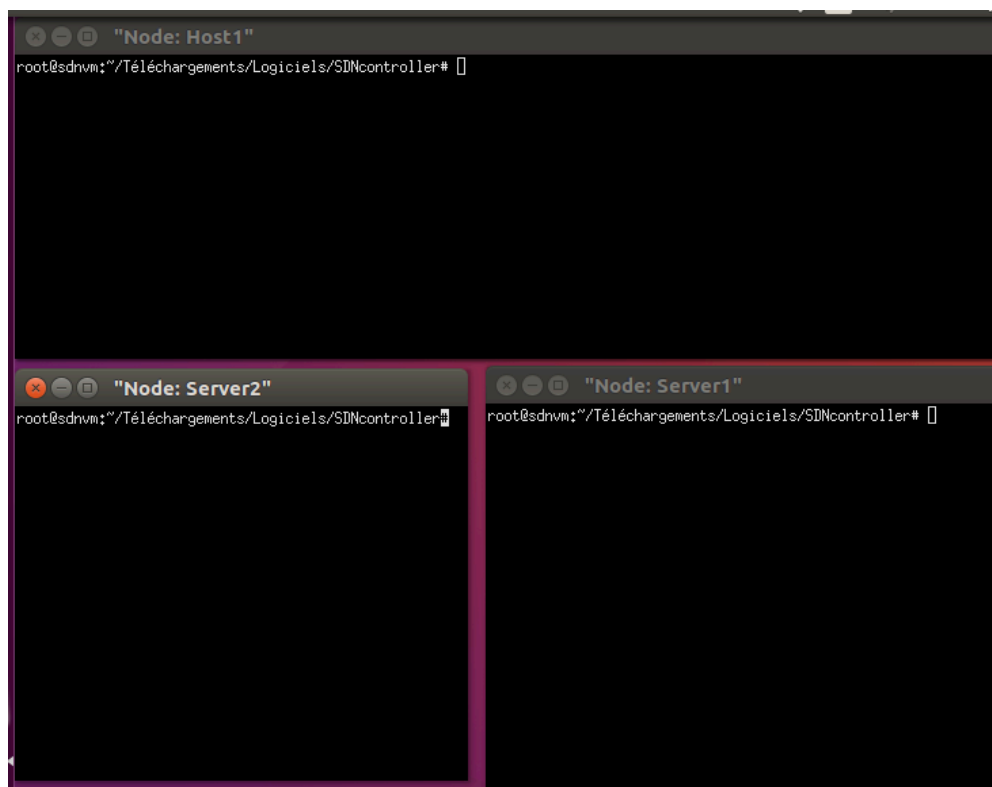
Ces commandes lancent un serveur HTTP simple sur les hôtes Server1 et Server2, écoutant sur le port 80, avec une exécution en arrière-plan grâce à l'opérateur &. Cette configuration a permis de simuler le fonctionnement de la topologie réseau avec des serveurs Web accessibles pour les tests ultérieurs.

B) Redirection transparente de paquets IP :

Dans le cadre de ce projet, une émulation de console a été effectuée sur l'Hôte 1, le Serveur 1 et le Serveur 2 à l'aide de xterm, afin d'interagir individuellement avec chaque entité de la topologie. Les commandes utilisées dans Mininet pour lancer ces consoles sont les suivantes :

```
containernet> xterm Server1  
containernet> xterm Server2  
containernet> xterm Host1
```

on obtient :



Ensuite, pour analyser le trafic réseau traversant les interfaces des Serveurs 1 et 2, l'outil tcpdump a été utilisé dans leurs consoles respectives. Les commandes

exécutées étaient :

sudo tcpdump -i Server1-eth0 icmp sur le Serveur 1 et sudo tcpdump -i Server2-eth0 icmp sur le Serveur 2.

"Node: Server2"	"Node: Server1"
<pre>root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# sudo tcpdump -i Server2-eth0 icmp tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on Server2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes []</pre>	<pre>root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# sudo tcpdump -i Server1-eth0 icmp tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on Server1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes []</pre>

Ces commandes permettent de capturer et d'afficher les paquets ICMP circulant sur leurs interfaces réseau Server1-eth0 et Server2-eth0. Lors de l'observation, le trafic capturé est principalement constitué de paquets LLDP (Link Layer Discovery Protocol) envoyés par le contrôleur SDN pour découvrir la topologie réseau.

"Node: Server1"	"Node: Host1"
<pre>tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on Server1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes 15:12:10.638347 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7374, length 64 15:12:10.638363 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7374, length 64 15:12:11.529380 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7374, length 64 15:12:11.529418 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7374, length 64 15:12:12.529345 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7374, length 64 15:12:12.529358 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7374, length 64 15:12:13.52626 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7374, length 64 15:12:13.52662 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7374, length 64 15:12:14.576272 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7374, length 64 15:12:14.576311 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7374, length 64 []</pre>	<pre>root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# ping 10.0.0.2 PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data: 64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=118 ms 64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.423 ms 64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.037 ms 64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.097 ms 64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.103 ms ^C --- 10.0.0.2 ping statistics --- 5 packets transmitted, 5 received, 0% packet loss, time 4048ms rtt min/avg/max/mdev = 0.037/23.793/118.305/47.256 ms root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller#</pre>

"Node: Server2"	"Node: Host1"
<pre>h 64 15:12:58.159057 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 4, length 64 15:12:59.184179 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 7723, seq 5, length 64 15:12:59.184190 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 5, length 64 15:13:00.209653 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 7723, seq 6, length 64 15:13:00.209664 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 6, length 64 15:13:01.231124 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 7723, seq 7, length 64 15:13:01.231137 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 7, length 64 15:13:02.255075 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 7723, seq 8, length 64 15:13:02.255087 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 8, length 64 15:13:03.279098 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 7723, seq 9, length 64 15:13:03.279111 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 7723, seq 9, length 64</pre>	<pre>root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# ping 10.0.0.2 PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data: 64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=118 ms 64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.423 ms 64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.037 ms 64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.097 ms 64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.103 ms ^C --- 10.0.0.2 ping statistics --- 5 packets transmitted, 5 received, 0% packet loss, time 4048ms rtt min/avg/max/mdev = 0.037/23.793/118.305/47.256 ms root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# ping 10.0.0.3 PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data: 64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=14.9 ms 64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.153 ms 64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.029 ms 64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.028 ms 64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.032 ms 64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.082 ms 64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.080 ms 64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.036 ms 64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=0.041 ms</pre>

Enfin, depuis la console xterm de l'Hôte 1, des commandes de ping ont été exécutées vers le Serveur 1 et le Serveur 2. Les résultats observés montrent que les paquets ICMP générés par le ping sont visibles dans les captures effectuées par tcpdump sur les Serveurs 1 et 2, confirmant ainsi la connectivité entre les hôtes et les serveurs, ainsi que le bon fonctionnement de la topologie.

Ensuite, une règle de flux a été configurée et installée sur le switch SDN afin de rediriger les paquets IP de manière transparente de Host1 vers Server2, tout en préservant l'apparence d'une communication directe entre Host1 et Server1. L'identifiant du switch (00:00:00:00:00:00:01) a été récupéré à l'aide de la commande suivante :

```
sdnvm@sdnvm: ~/Téléchargements/Logiciels/SDNcontroller
sdnvm@sdnvm:~$ cd Téléchargements/Logiciels/SDNcontroller/
sdnvm@sdnvm:~/Téléchargements/Logiciels/SDNcontroller$ sudo ovs-ofctl show Switc
h1
[sudo] Mot de passe de sdnvm :
OFPST_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
```

ou via l'interface Web du contrôleur SDN à l'url 10.0.2.15:8080/ui/index.html.



L'action spécifiée consiste à réécrire les adresses de destination avec l'adresse MAC de Server2 et l'adresse IP de Server2, et les paquets sont dirigés vers le port de sortie correspondant à Server2 (port 3). La règle de flux a été installée à l'aide de la commande suivante :

```
sdnvm@sdnvm:~/Téléchargements/Logiciels/SDNcontroller$ curl -X POST -d '{"switc
h":"1", "name":"f2", "priority":"36000", "in_port":"1", "eth_type":"0x0800", "et
h_src":"00:00:00:00:00:01", "ipv4_src":"10.0.0.1", "eth_dst":"00:00:00:00:00:02",
"ipv4_dst":"10.0.0.2", "actions":"set_field=eth_dst->00:00:00:00:00:03, set_fie
ld=ipv4_dst->10.0.0.3, output=3}' http://10.0.2.15:8080/wm/staticflowpusher/json
```

Depuis la console de Host1, un ping a été exécuté vers Server1 à l'aide de la commande suivante :

```
"Node: Server1"
root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# sudo tcpdump -i Server1-e
th0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on Server1-eth0, link-type EN10MB (Ethernet), capture size 262144 byte
s
[]

"Node: Server2"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on Server2-eth0, link-type EN10MB (Ethernet), capture size 262144 byte
s
15:45:42.980215 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 13665, seq 1, leng
th 64
15:45:42.980231 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 13665, seq 1, lengt
h 64
15:45:43.981770 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 13665, seq 2, leng
th 64
15:45:43.981804 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 13665, seq 2, lengt
h 64
15:45:44.983956 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 13665, seq 3, leng
th 64
15:45:44.983985 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 13665, seq 3, lengt
h 64
15:45:45.987941 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 13665, seq 4, leng
th 64
15:45:45.987970 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 13665, seq 4, lengt
h 64
15:45:46.991155 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 13665, seq 5, leng
th 64
15:45:46.991186 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 13665, seq 5, lengt
h 64
```

```
root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# ping 10.0.0,2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.43 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.421 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.093 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/ndev = 0.093/1.827/8.431/3.304 ms
root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# ping 10.0,0,2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=3.21 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.300 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.083 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.135 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4010ms
rtt min/avg/max/ndev = 0.088/0.773/3.217/1.221 ms
root@sdnvm:~/Téléchargements/Logiciels/SDNcontroller# []
```


Les observations montrent que les paquets ICMP sont interceptés par le switch, redirigés vers Server2, et capturés par tcpdump sur Server2, confirmant que Server1 ne reçoit pas les paquets ICMP. La redirection est effective, car les adresses MAC et IP sont réécrites, permettant à Host1 de croire qu'une communication directe avec Server1 a lieu.

Pour rendre la redirection pleinement fonctionnelle dans les deux sens (réponse de Server2 à Host1), une règle supplémentaire a été ajoutée pour gérer les paquets en retour :

```
sdnvm@sdnvm:~$ curl -X POST -d '{"switch":"1", "name":"f3", "priority":"36000", "in_port":"3", "eth_type":"0x0800", "eth_src":"00:00:00:00:00:03", "ipv4_src":"10.0.0.3", "eth_dst":"00:00:00:00:00:01", "ipv4_dst":"10.0.0.1", "actions":"set_field=eth_dst->00:00:00:00:00:02,set_field=ipv4_dst->10.0.0.2,output=1"}' http://10.0.2.15:8080/wm/staticflowpusher/json
```

La redirection transparente a été mise en place et validée avec succès, démontrant l'efficacité de l'approche SDN pour contrôler dynamiquement le trafic réseau et adapter les flux en fonction des besoins.

Partie 3 : Intégration de modules à Floodlight

L'objectif de cette partie était d'acquérir les compétences nécessaires pour ajouter, désactiver et supprimer des modules dans Floodlight, ainsi que de générer une version personnalisée du contrôleur. Cette étape inclut également le développement de nouveaux modules offrant des services spécifiques via des API Java et REST.

Les modules proposés pour le développement incluaient un redirecteur de paquets IP dans le tempsnet, permettant d'alterner l'envoi des paquets entre deux serveurs selon des périodes définies, un redirecteur de paquets IP à la demande offrant à l'utilisateur la possibilité de rediriger les paquets vers un serveur spécifique via un appel REST, un moniteur de flux à la demande capable de générer une matrice de trafic par protocole (HTTP, UDP, ICMP, etc.) et d'exposer ces données via des appels REST, ainsi que tout autre module défini selon des besoins spécifiques.

La méthode pour ajouter un module consistait à créer un package Java dans le projet *sdncontroller* sous le dossier *src/main/java*, avec un nom correspondant au module, comme *net.tpsdn.mymod*. Une interface Java exposant les méthodes du module devait ensuite être définie, héritant de *IFloodlightService* pour permettre son intégration avec d'autres modules. Enfin, une classe principale devait être développée pour implémenter *IFloodlightModule* et l'interface personnalisée du module. Cette classe devait inclure les méthodes clés : *getModuleDependencies* pour spécifier les dépendances, *init* pour initialiser les variables nécessaires, et *startUp* pour démarrer la logique du module.

L'intégration du module dans Floodlight nécessitait de modifier deux fichiers essentiels : *META-INF/services*, utilisé pour enregistrer le module, et

floodlightdefault.properties, configuré pour charger le module automatiquement au démarrage du contrôleur.

Malheureusement, par manque de temps, cette partie n'a pas pu être réalisée dans le cadre du projet. Cependant, le principe et la méthodologie ont été bien compris, et les bases nécessaires pour développer et intégrer un module dans Floodlight sont maîtrisées.

Conclusion

En conclusion, ce projet a permis de développer une compréhension approfondie des réseaux définis par logiciel (SDN) en explorant des outils tels que Floodlight et Mininet/Containernet. Les différentes étapes réalisées, notamment la configuration des règles de flux, l'émulation de réseaux SDN, l'utilisation de l'interface Web du contrôleur, et la redirection transparente de paquets IP, ont démontré l'efficacité et la flexibilité des solutions SDN pour la gestion dynamique des réseaux.

Bien que certaines parties, comme l'intégration de modules spécifiques à Floodlight, n'aient pas pu être finalisées par manque de temps, les méthodologies et concepts nécessaires pour accomplir ces tâches ont été bien assimilés. Cette expérience a offert une base solide pour aborder de futurs projets impliquant la personnalisation et l'optimisation de solutions SDN. Elle souligne également l'importance de l'adaptabilité et de la rigueur dans la gestion des réseaux modernes, tout en ouvrant des perspectives intéressantes pour approfondir les recherches dans ce domaine.