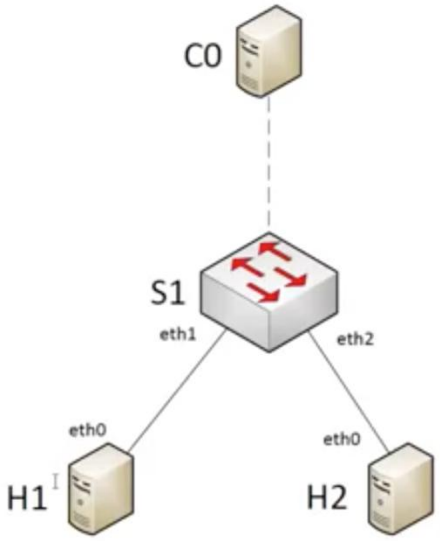
Final Project

1. ARP spoofing

You have to google search, visit Ryu documentation web-page, and especially the description of the OpenFlow v1.0 messages and structures, in order to develop an OpenFlow controller using the Ryu framework.

In particular, you have to develop a **switch replying to ARP requests**, by pushing the appropriate ARP replies to the network. For example, in the following figure,



the ARP requests produced by H1 for learning the MAC address of the H2's IP address, will be replied by the switch by producing the same ARP replies as H2 would do, if the packet was forwarded to it. **The switch will not forward the ARP request of H1 to H2, but will reply itself by producing the corresponding ARP reply.**

Notes

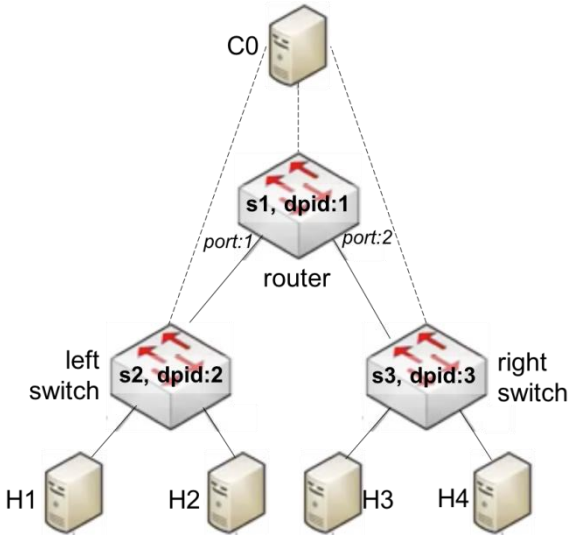
- Use the parameter *--mac* when you create the mininet environment, which gives serial and known MAC addresses to the hosts, starting from 00:00:...:01, 00:00:...:02, etc.
- The command 'arp -n' shows the ARP table of a host. For example, running in the mininet CLI the command 'h1 arp -n', the student will see the ARP table of h1.
- Add an *if* statement early in your code to return in packet-in, when the triggering packet has Ethertype equal to 0x86dd.

2. Static routing

Next, you have to extend your previous work to create a **static router** that interconnects two switches. You should run the script mininet-router.py, by copying this file to your working directory and executing:

```
root@mininet-vm:~# chmod 777 mininet-router.py
root@mininet-vm:~# ./mininet-router.py
```

The following Mininet topology is now deployed. The dashed lines represent the connections to the controller C0.



Left and right switches have datapath ids 0x2 and 0x3 respectively, and create separate LANs using IP networks **that are known to the router**. Let's say that the left LAN uses the IP network 192.168.1.0/24, while the right one uses the IP network 192.168.2.0/24. The router has datapath id 0x1 and behaves as a gateway for each LAN, having the IP addresses 192.168.1.1 and 192.168.2.1 at each of the two interfaces connecting to the two LANs. Extend the Ryu description of ryu-router-frame.py to control the whole network, creating two switches and one router. You have to fill in the code where is marked, in a way that the router:

- **replies to the ARP requests** for the IP addresses 192.168.1.1 and 192.168.2.1 of its own interfaces, assuming that the MAC addresses of its interfaces are 00:00:00:00:01:01 and 00:00:00:00:02:01 respectively,
- setups flows **reactively** and forwards packets from one LAN to the other, by matching the packets with their **destination IP addresses** and **changing their Ethernet headers** before forwarding.

Notes:

You already have examples of Match structure (*in_port & dl_dst*) and Actions list (single-item list *[OFPActionOutput(out_port)]*) from ryu-router-frame.py, which you should now change and

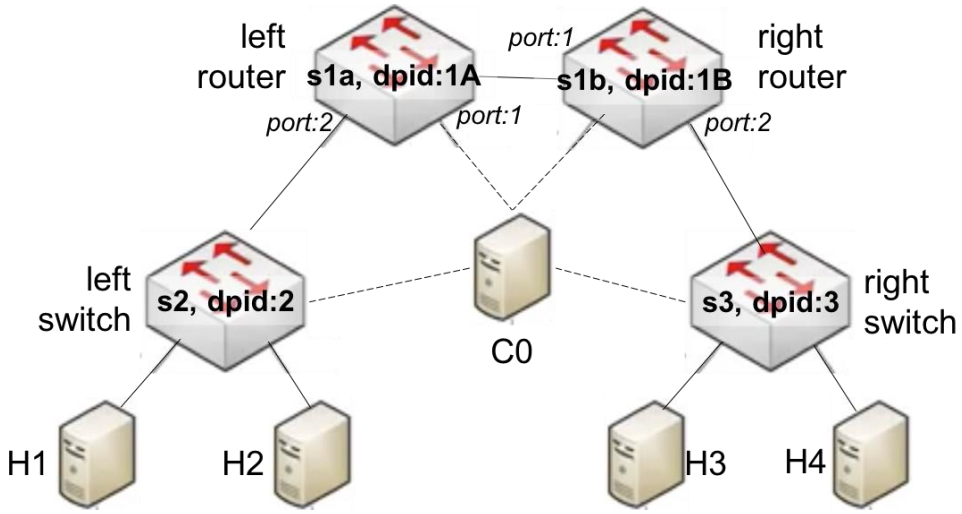
extend reading the sections Flow Match Structure and Action Structures respectively, from the given tutorial OpenFlow v1.0 messages and structures.

3. Static routing with two routers

Now you have to extend your previous work to create **two static routers** that interconnect two LANs. You should run the script mininet-router-two.py, by copying this file to your working directory and executing:

```
root@mininet-vm:~# chmod 777 mininet-router-two.py
root@mininet-vm:~# ./mininet-router-two.py
```

The following Mininet topology is now deployed.



Left and right switches have datapath ids 0x2 and 0x3 respectively, and create separate LANs using IP networks **that are known to the routers**. Let's say that the left LAN uses the IP network 192.168.1.0/24, while the right one uses the IP network 192.168.2.0/24. The left router has datapath id 0x1A and is the gateway for the left LAN, having the IP address 192.168.1.1 at its interface connected to the left LAN. The right router has datapath id 0x1B and is the gateway for the right LAN, having the IP address 192.168.2.1 at its corresponding interface connected to the right LAN. Have in mind that the left and right interfaces of the left router have the MAC addresses 00:00:00:00:01:01 and 00:00:00:00:03:01 respectively, while the left and right interfaces of the right router have the MAC addresses 00:00:00:00:03:02 and 00:00:00:00:02:01 respectively.

Extend the Ryu description of ryu-router-two-frame.py to control the whole network, creating two switches and two routers. You have to fill in the code, where is asked, in a way that each router:

- **replies to the ARP request** for the IP address 192.168.1.1 or 192.168.2.1 of its own interface,

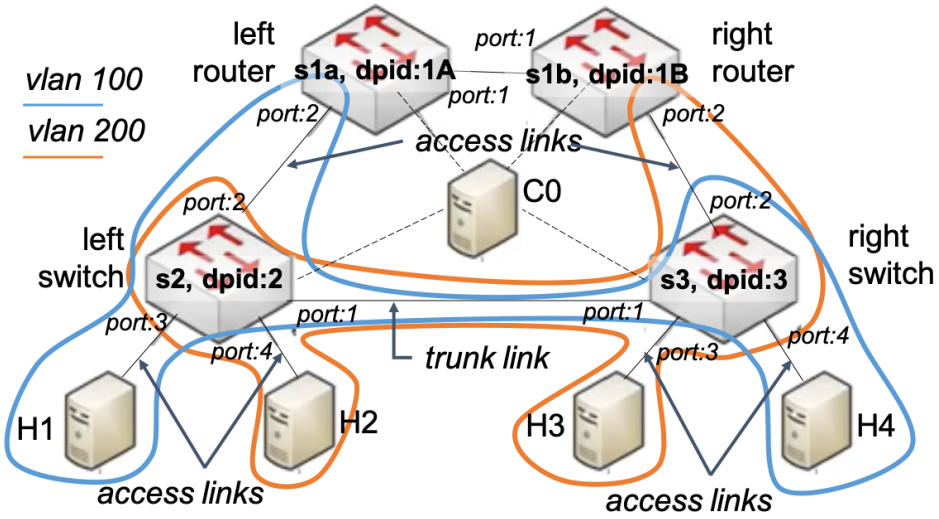
- setups flows **reactively** and forwards packets from one LAN to the other, by matching the packets with their **destination IP addresses** and **changing their Ethernet headers** before forwarding. The forwarding between the routers should be based on flows matching **IP network addresses**, not IP host addresses (use network mask).

4. VLAN with OpenFlow

Create a network with two interconnected switches, divided into two virtual LANs with use of the VLAN technology, which are connected through two routers. You should run the script mininet-router-vlan.py, by copying this file to your working directory and executing

```
root@mininet-vm:~# chmod 777 mininet-router-vlan.py
root@mininet-vm:~# ./mininet-router-vlan.py
```

which creates in Mininet the following topology. The dashed lines represent the connections to the controller C0.



Left and right switches have datapath ids 0x2 and 0x3 respectively, and create two VLANs with ids 100 and 200, which use IP networks **known to the routers**. Let's say that VLAN 100 uses the IP network 192.168.1.0/24, while VLAN 200 uses the IP network 192.168.2.0/24. The left router has datapath id 0x1A and behaves as a gateway for VLAN 100, having the IP address 192.168.1.1 at its interface connected to the left switch and participating in VLAN 100. The right router has datapath id 0x1B and is the gateway for VLAN 200, having the IP address 192.168.2.1 at its corresponding interface connected to the right switch and participating in VLAN 200. When packets pass through the **access links**, they are **not VLAN encapsulated**, while the trunk link is the one that packets are **VLAN encapsulated**. Have also in mind that the left and right interfaces of the left router have the MAC addresses 00:00:00:00:01:01 and 00:00:00:00:03:01 respectively, while the left and right interfaces of the right router have the MAC addresses 00:00:00:00:03:02 and 00:00:00:00:02:01 respectively.

Extend your controller from the previous section (*Static routing with two routers*), in a way that each router does the same job with before, and each switch handles the packets and **reactively** configures flows for:

- **receiving packets from its access ports** and forwarding them according to the normal L2 functionality. If these packets are to be forwarded to a trunk link, they are **encapsulated with the known VLAN id** (e.g. left switch encapsulates packets coming from port 3 and pushes to port 1, with VLAN id 100).
- **receiving packets from its trunk ports** and forwarding them according to the normal L2 functionality. If these packets are to be forwarded to an access link, they are **decapsulated from the VLAN header**.

Have in mind that packets exchanged between two hosts of the same VLAN, should pass through the trunk link if the hosts are attached to different switches. On the other hand, if the hosts belong to different VLANs, then their exchanged packets should pass through the routers.

Notes:

You have also to implement the following functionalities:

1. Both routers must reply with an **ICMP type 3, "Destination Host Unreachable"** packet, when they receive a packet that is destined to an unknown IP address, meaning an IP address that does not belong to the IP networks 192.168.1.0/24 and 192.168.2.0/24.
2. Extend your mininet topology to include also an extra link between the routers, using their port 4, and extend your controller to use this link for **proactive forwarding of the high-priority traffic** (let's say the traffic with non-zero ToS) **between the routers**. Use the parameter -S 8 of iperf to produce non-zero ToS (high-priority) packets. Have in mind that the new interfaces of the left and right routers have the MAC addresses 00:00:00:00:05:01 and 00:00:00:00:05:02 respectively.

Your code should be executable. Please put your 4-digit AEM (without leading 0s) as a **prefix** of your file name, **write it in the first line of the script using comments,** and **submit it**.

****Deadline:** Sunday 30/06/2024**