UNIVERSITY OF PISA

Advanced Network Architectures and Wireless Systems

Year 2019/20

Centralized Virtual Redundancy Protocol

using a Floodlight OpenFlow controller

GERARDO ALVARO

LEONARDO FONTANELLI

RICCARDO POLINI

Introduction

The goal of this project is to define and implement a Floodlight Controller for SDN able to manage packets on a SDN switch (SS) and to route them from clients in Network A to servers in Network B through one gateway (either R1 or R2), as shown in Figure 1. Switch LS is considered as a legacy switch.

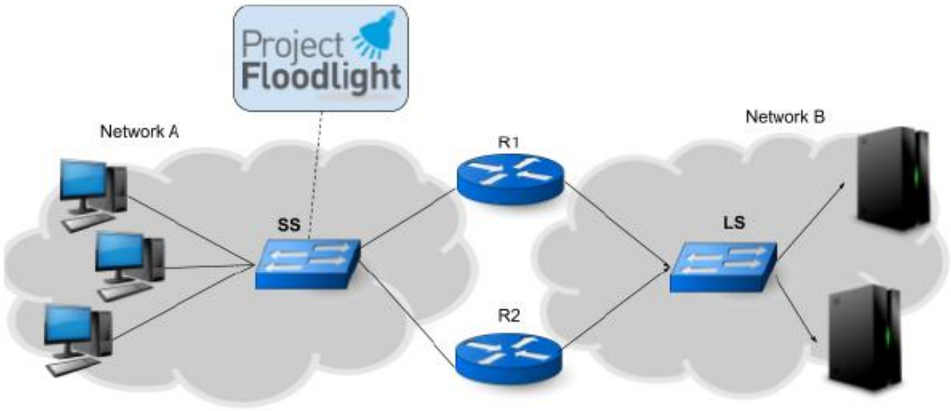


Figure 1: View of the global network

More specifically, at any time only one router operates as the gateway (Master), while the other one is available as a backup and will operate just in case of Master failure. Clients will be unaware of which router is acting as gateway. This will be achieved by masking the router's addresses with a virtual one.

Repository Structure

The project repository is organized as follows:

- **`java:`** contains 3 Java files (both controllers will be discussed later in detail):
    - **`ARPController:`** main file that handles ARP and ICMP packets.
    - **`TController:`** handles the router Redundancy protocol. Checks whether the master router is offline and, in case, elects a new master router.
    - **`Parameters:`** contains configurable parameters.
- **`python`:** contains the following Python files:
    - **`net`:** mininet script that builds the network configuration as shown in Figure 1.
    - **`router:`** contains the router operational code.

Redundancy Protocol

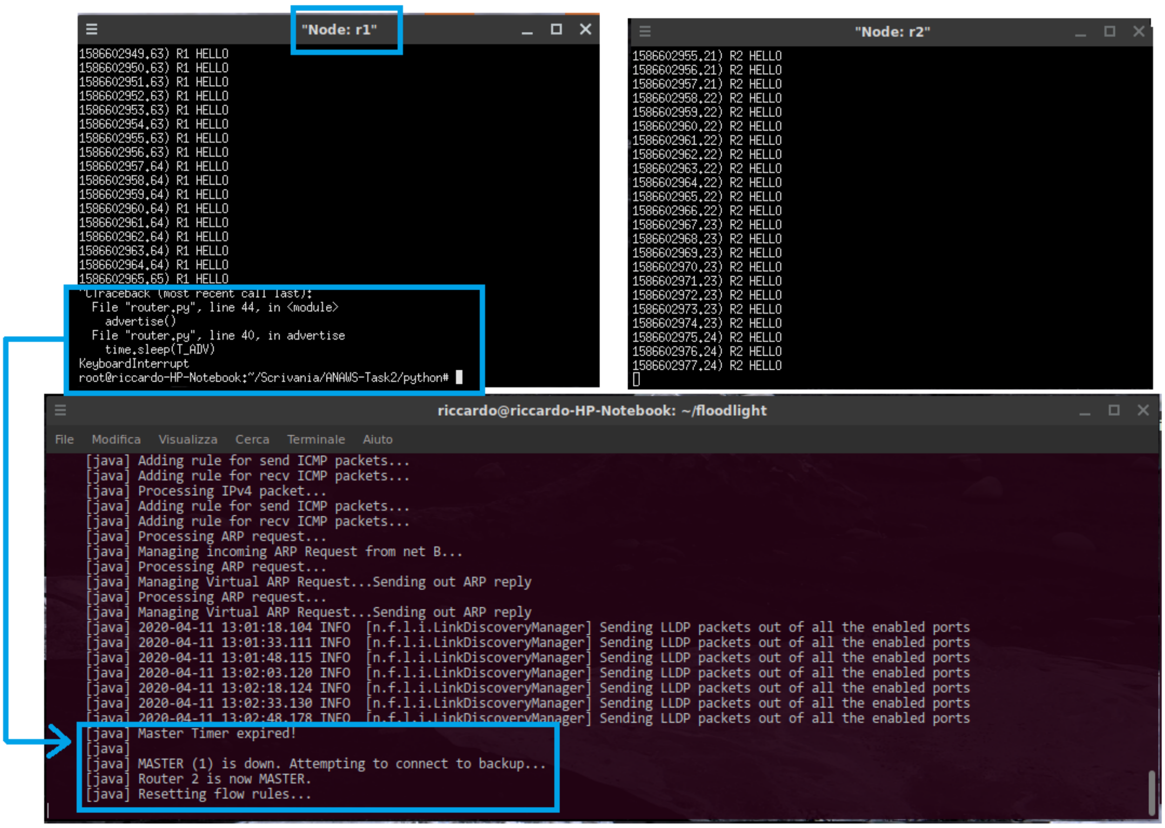
Guarantees the functionality of the network if the master router goes down (and a backup is available). Implemented in the **`TController.java`** file.

By default, and for a matter of simplicity, router R1 is the initial master and R2 is the backup. Each router sends a hello message consisting in the **`ROUTER_ID`**, every second on the UDP port 8787.

The **`TController`** module listens on port 8787 for router advertisements. When it receives one, it saves in a Hash Map the timestamp of arrival of the message, for that given router id.

The module also runs a timer which expires every 3 seconds. Every time an advertisement is received from the master router, the timer is reset. If the timer expires without having received an advertisement from the master router in the last 3 seconds, the router is considered offline and the backup router is elected as the new master (if available).

If no backup is available, the program is closed.



In the above figure we can see that when R1 (master) is stopped, after the timer (3 seconds) expires, the backup router R2 is elected. If we restart R1, it will be set as backup router and will become master again only if R2 is stopped.

Address Plan

The following table shows the addresses we chose for the hosts of the network.

Host	IP
R1	10.0.2.1
R2	10.0.2.2
a-h1	10.0.2.3
a-h2	10.0.2.4
a-h3	10.0.2.5
b-h4	10.0.3.3
b-h5	10.0.3.4

As for the virtual addresses, we chose VIRTUAL_IP as **`10.0.2.254`** (last address of network A) and VIRTUAL_MAC as **`00:00:5E:00:01:01`** as suggested by RFC5798: *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6*.

Virtual Address Masking

The *n* routers that lay in between network A and B are seen as one entity having a VIRTUAL_IP and a VIRTUAL_MAC. To implement this, the controller has to modify all packets involved with the routers. This means to replace the router's real addresses with the virtual ones.

When a packet arrives at the Floodlight controller, there can be two cases:

1. **ARP packet**
    a. **eth_source = REAL:** a host from network B is trying to communicate with network A. The gateway router sends an ARP Request to discover the address of the corresponding host in network A. The controller intercepts an ARP Request with the router's real MAC as *ethernet source* and sets the virtual addresses as sender addresses. A rule is saved in order to perform the same operation when the same conditions are intercepted again.
    The controller also produces the inverse rule for the case when a host replies to the ARP Request. This rule consists in intercepting the host's ARP Reply and inserting the <u>real</u> router's addresses (host will reply to the virtual ones).

    b. **IP_destination = VIRTUAL:** the controller intercepts a broadcast ARP Request from network A which is trying to discover the VIRTUAL_MAC associated to the VIRTUAL_IP of the router. The controller builds and ARP Reply packet specifying the VIRTUAL_MAC and sends it back to the host.

2. **ICMP packet**
    a. **eth_destination = VIRTUAL:** the controller intercepts an ICMP packet from network A destined to the VIRTUAL_MAC and replaces it with the <u>real</u> router's MAC address. ICMP *source* and *destination* will still be host_A and host_B. A rule is saved in order to perform the same operation when the same conditions are intercepted again.

    b. **eth_source = REAL:** the controller intercepts an ICMP packet coming from router R2 (default gateway for network B). It sets VIRTUAL_MAC as the *ethernet source*.

Execution example

To test the software, the first thing to do is to launch the python script that generates the network by means of *mininet*.

```
sudo python net.py
```

Once inside the *mininet* console, we need to launch the two routers:

```
mininet> xterm r1 && xterm r2
```

Then we launch each router:

```
r1> python r1.py 1
```

```
r2> python r2.py 2
```

At this point we can launch floodlight and perform some tests.

ARP/ping example

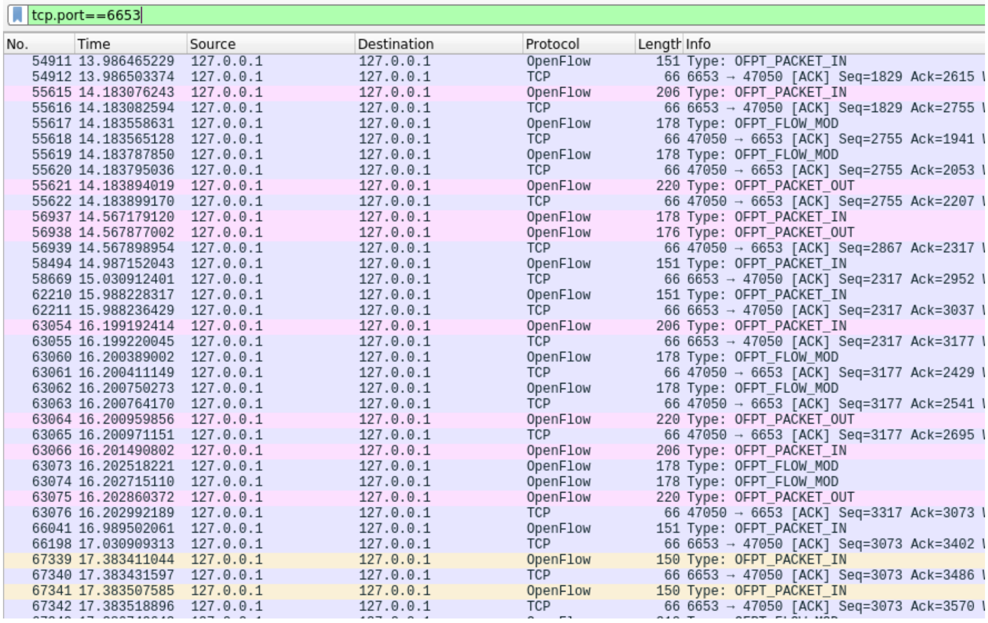
We issued a ping request from h1 to h4 and tracked it with Wireshark on the interface between h1 and the switch.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:00:00_00:00:a1	Broadcast	ARP	42	Who has 10.0.2.254? Tell 10.0.2.3
2	0.006530052	IETF-VRRP-VRID_01	00:00:00_00:00:a1	ARP	42	10.0.2.254 is at 00:00:5e:00:01:01
3	0.006553922	10.0.2.3	10.0.3.3	ICMP	98	Echo (ping) request id=0x1aa4, seq=1/256, ttl=64 (reply in 6)
4	0.100785781	IETF-VRRP-VRID_01	Broadcast	ARP	42	Who has 10.0.2.3? Tell 10.0.2.254
5	0.100798148	00:00:00_00:00:a1	IETF-VRRP-VRID_01	ARP	42	10.0.2.3 is at 00:00:00:00:00:a1
6	0.101094205	10.0.3.3	10.0.2.3	ICMP	98	Echo (ping) reply id=0x1aa4, seq=1/256, ttl=63 (request in 3)
7	0.997243887	10.0.2.2	10.0.2.255	UDP	43	8787 → 8787 Len=1
8	1.001389347	10.0.2.3	10.0.3.3	ICMP	98	Echo (ping) request id=0x1aa4, seq=2/512, ttl=64 (reply in 9)
9	1.001900751	10.0.3.3	10.0.2.3	ICMP	98	Echo (ping) reply id=0x1aa4, seq=2/512, ttl=63 (request in 8)
10	1.998441835	10.0.2.2	10.0.2.255	UDP	43	8787 → 8787 Len=1
11	2.002468193	10.0.2.3	10.0.3.3	ICMP	98	Echo (ping) request id=0x1aa4, seq=3/768, ttl=64 (reply in 12)
12	2.002617014	10.0.3.3	10.0.2.3	ICMP	98	Echo (ping) reply id=0x1aa4, seq=3/768, ttl=63 (request in 11)
13	3.030401573	10.0.2.3	10.0.3.3	ICMP	98	Echo (ping) request id=0x1aa4, seq=4/1024, ttl=64 (reply in 14)
14	3.041746812	10.0.3.3	10.0.2.3	ICMP	98	Echo (ping) reply id=0x1aa4, seq=4/1024, ttl=63 (request in 13

In message 1, h1 broadcasts an ARP Request to figure out the MAC corresponding to the `10.0.2.254` address (VIRTUAL IP). h1 receives an ARP Reply (produced by the controller) in which there is the VIRTUAL MAC (`00:00:5e:00:01:01`). At this point the ICMP requests/reply start working flawlessly.

Controller Rules example

What happens on the controller side? We can open Wireshark on the Loopback interface and apply **tcp.port==6653** as a filter. if we run the previous example, we will see the FLOW_MOD of the controller.



After each PACKET_IN there are 2 FLOW_MODs because we install a rule plus the reverse of that rule to allow for the response to flow correctly through the network.

In the image below we can see the contents of a FLOW_MOD packet. in this specific case the Match condition recognizes the VIRTUAL_MAC as the Ethernet Destination of a packet and installs the Action which consists in replacing the VIRTUAL_MAC with the current master router's MAC (in this case R1).