ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Ryu controller programming - Hands-on session

Chiara Grasselli

LAB. OF NETWORK PROGRAMMABILITY AND AUTOMATION - PROGRAMMABLE NETWORKING (A.Y. 2024/2025)

Useful tools

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Ryu documentation

Useful links:

- https://ryu.readthedocs.io/en/latest/developing.html
- https://osrg.github.io/ryu-book/en/html/packet_lib.html

Iperf tool

- Consult iperf doc: ***man* iperf**
  - Perform network throughput tests

- Iperf allows you to generate TCP and UDP traffic
  - By default, TCP traffic

- You need to run a server instance and a client instance

Iperf main options: server

- iperf server instance listen (by default) on port 5001
- iperf server side:
  - **-s** : indicates a running TCP Server, waiting for incoming connections
  - **-s -u** : indicates a running UDP Server
  - **-p <port_number>** : set server port to listen on/connect to port_number

Iperf main options: client

- iperf client side
  - **-c <server_IP_address>** : indicates a running TCP client connecting to TCP server IP address
  - **-c -u <server_IP_address>** : indicates a running UDP client contacting UDP server IP address
  - **-t <time>** : indicates the duration (in seconds) of time it sends TCP packet to the server
  - **-i <time>** : indicates the interval (in seconds) after which it prints the average bandwidth measured
  - **-t** and **–i** can be combined with **–u** option for the UDP version

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Iperf TCP instance example

- **iperf -s** : starts a TCP server instance
- **iperf -c 10.0.0.6 -t 200 -i 10** : starts a TCP client instance connecting to 10.0.0.6 TCP server, running for 200 seconds, and printing statistics every 10 seconds

Netcat

Netcat (nc) is a simple application that enables the use of the transport layer on top of IP

- **nc -l -n 192.168.1.1 7090**
  - Opens trasport port 7090 using the TCP protocol in LISTEN mode (waiting for a call)
- **nc -n 192.168.1.1 7090**
  - Calls host 192.168.1.1 on transport port 7090 using the TCP protocol
- Other option
  - **-u** : will force the use of the UDP protocol instead of TCP
  - **-p** : allows you to specify the source port to be used when conecting
  - **-s** : allows you to use a specific IP address in the messages

Ryu controller programming
Hands-on session

Netprog_ex1_dst_only.py

- Copy *Netprog_ex1_dst_only.py* inside the Ryu app folder
- Run *1switch_3host_ext_cntlr_no_ipv6.py* as Mininet topology
- Observe the controller behavior
    - Generate some traffic between hosts with ping
    - Check the traffic between switch and controller

  ...Anything weird?

- Check the Python code

- Compare with *Netprog_ex1_src_dst.py*
    - Run the controller and repeat the previous steps. What happens this time?

Implement the mirroring network function

- Use *1switch_3host_ext_cntlr.py* (or *1switch_3host_ext_cntlr_no_ipv6.py*) as Mininet topology
- Inside the Ryu app folder, make a copy of the simple switch 1.3 (*simple_switch_13.py*)
- Edit the code to implement a simple **mirroring policy**, for example: *all traffic coming from host1 and directed towards host2 is mirrored to host3*
- Note: this policy can be implemented in different ways. Start from the simplest version and then try to change it.

Implement the firewall network function

- Use *1switch_3host_ext_cntlr_no_ipv6.py* (or *1switch_3host_ext_cntlr.py*) as Mininet topology
- Inside the Ryu app folder, make a copy of the simple switch 1.3 (*simple_switch_13.py*)
- Edit the code to implement a **firewall policy**, for example: *icmp traffic from host2 to host3 is not allowed*

- Use the firewall as a playground and try to implement other firewall policy, e.g.:
  - TCP traffic directed to port 5001 from host1 to host3 is allowed
  - TCP traffic directed to port 5002 from host1 to host3 is not allowed