P7 configuration

The user needs to define the topology and links configuration. It is possible to define all the network elements that will be emulated with the corresponding parameters.

The topology generation template is defined in the *main.py* file. Using this script, are generated the necessary files of P7 (Figure 1).

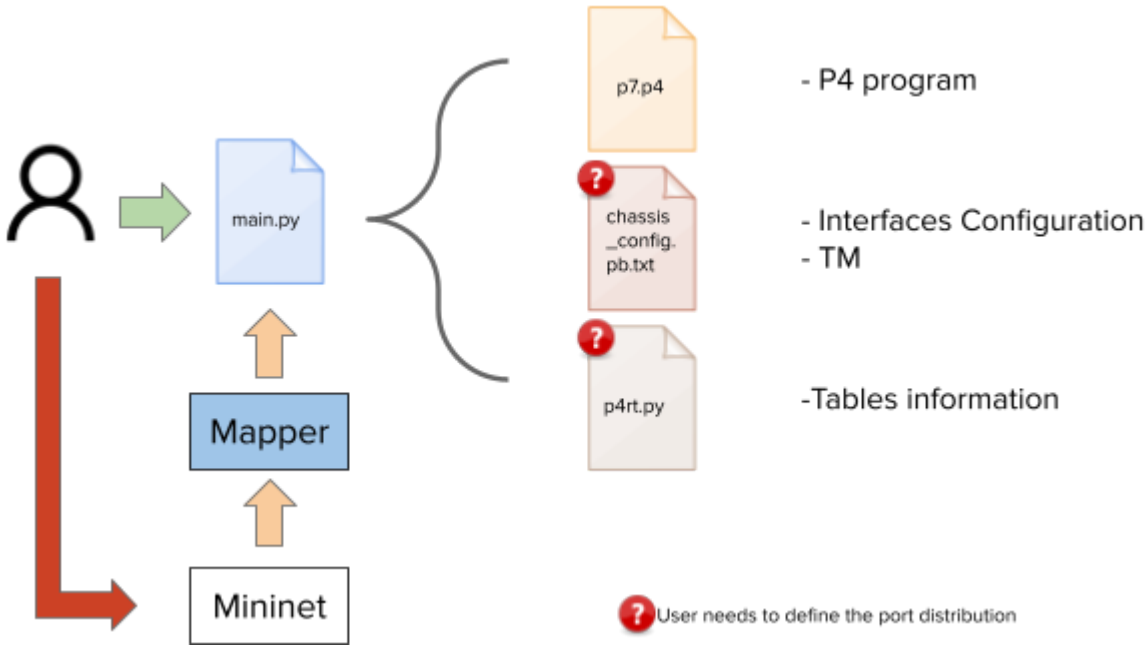

Figure 1

The output of the topology generation script is three files:

- **P4 program**
  - Contains the P4 code structure, including the emulated links and devices.
  - From the main script are mapped the principal parameters for the network emulation.
- **Interfaces configuration**
  - To successfully run the switch, the chassis configuration must be set.
  - This file contains all the port configurations, including the speed, operational mode, and others (see section Chassis configuration).
- **Tables information**
  - This file contains the information of the P4 tables that will be configured.

Topology Generation Script

A template of the Topology Generation Script is available in */home/admin12/p7/p7_rnp/main.py.*

First, we need to initialize the generator object:

```
generate_p7 = generator('main')
```

We can define the IP and port of Stratum with *addstratum*. If it runs in the local device, it can be set with the management IP and default port 9559.

```
generate_p7.addstratum("192.168.110.238:9559")
```

The recirculation port is defined by default in port 68. But, it is possible to define other ports as loopback ports (refer to Tofino architecture documentation):

```
generate_p7.addrec_port(68)
```

With *addswitch,* we can define the number of internal devices that we will emulate. This item is not mandatory. If we do not define any switch, the emulated network will represent a link (Figure 2).

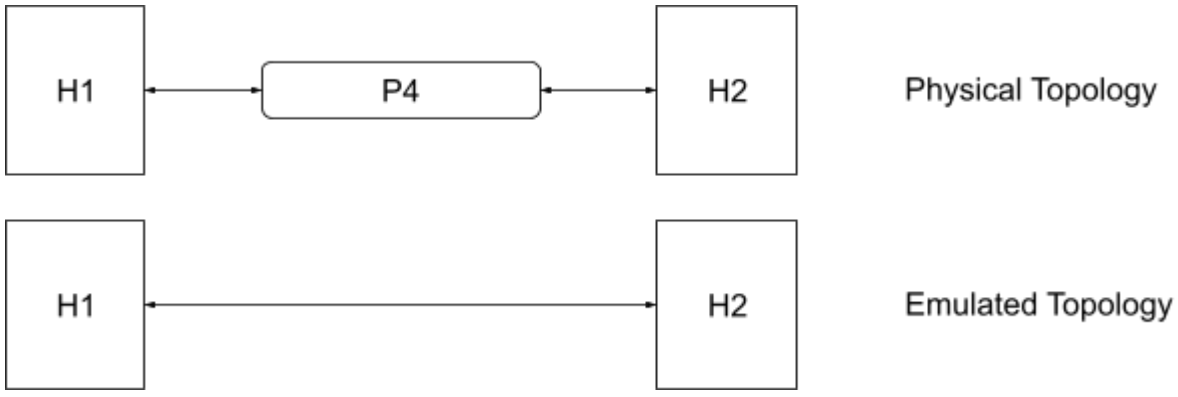

Figure 2

```
generate_p7.addswitch("sw1")
```

With *addhost,* we define the hosts with a physical connection with the switch. It is necessary to define the characteristics of the port.

```
addhost(name, port, D_P, speed_bps, AU, FEC, vlan)
```

where:

- **name:** name of the host.
- **port:** the physical port number.
- **D_P:** the id of the port.
- **speed_bps:** the speed in bps (e.g., 100000000000 -> 100Gbps).
- **AU:** automatic negotiation True/False.
- **FEC:** FEC mode enables True/False.
- **vlan:** vlan id for P7 processing.

```
generate_p7.addhost("h1",19, 20, 100000000000, "False", "False", 1920)
generate_p7.addhost("h2",20, 28, 100000000000, "False", "False", 1920)
```

With addlink, we define the internal links in the network. It is necessary to define the emulated characteristic of the link.

```
addlink(node1, node2, bw, pkt_loss, latency)
```

where:

- **node1:** name of the first node.
- **node2:** name of the second node
- **bw:** bandwidth limit in bps (e.g., 1000000000 -> 1Gbps). Bw is considered just for the first defined link.
- **pkt_loss:** percentage of packet loss.
- **latency:** latency of the network in milliseconds. The value of the link will be set in both sides (bidirectional latency). RTT will be two times this value.

```
generate_p7.addlink("h1","sw1", 100000000000, 0, 5)
generate_p7.addlink("sw1","h2", 100000000000, 0, 5)
```

Here is an example of a single link (no switch) definition:

```
generate_p7.addlink("h1","h2", 100000000000, 0, 5)
```

In parallel of the P7 processing, it is possible to define VLANS that are not processing by P7 and are directly forwarder.

Using *addvlan_port* and a similar structure of *addhost* we define aditional VLAN.

```
addvlan_port(port,D_P,speed_bps,AU,FEC)
```

where:

- **port:** the physical port number.

- **D_P:** the id of the port.
- **speed_bps:** the speed in bps (e.g., 100000000000 -> 100Gbps).
- **AU:** automatic negotiation True/False.
- **FEC:** FEC mode enables True/False.

```
generate_p7.addvlan_port(7, 180, 100000000000, "False", "False")
generate_p7.addvlan_port(16, 0, 100000000000, "False", "False")
```

In addition of addind a VLAN port, it is necessary to ad the VLAN link:

```
addvlan_link(D_P1, D_P2, vlan)
```

where:

- **D_P1:** First port id.
- **D_P2:** Second port id.
- **vlan:** VLAN id.

```
generate_p7.addvlan_link(180,0, 716)
```

Finally, we generate the P7 files.

```
generate_p7.generate_chassis()
generate_p7.generate_p4rt()
generate_p7.generate_p4code()
```

Running P7

After setting the topology generation script, we run the file to create the necessary P7 files.

```
# cd /home/admin12/p7/p7_rnp
# python main.py
```

Now, we need to set the files in the correct location.

```
# ./set_files.sh
```

We can start Stratum. First, we set the generated CHASSIS_CONFIG file:

```
# cd /home/admin12/p7
# export CHASSIS_CONFIG="/home/admin12/p7/p7_rnp/files/chassis_config.pb.txt"
# ./start-stratum.sh
```

If we do not close or change the terminal, we do not need to export CHASSIS_CONFIG again.

In a different terminal, we can compile the P4 code and start the P4 runtime connection to fill the tables and send the P4.

With all the files in the correct place, we can compile the P4 program.

```
# cd p4src
# ./genPipeConf.sh p7_default
```

```
# ./p4rt/start_p4runtime.sh
```

With all set, we can start testing the network.