Softwarized And Virtualized Mobile Networks

Carlin Nicola, Mich Edoardo, Moletta Davide

On Demand SDN Slices

Scope:

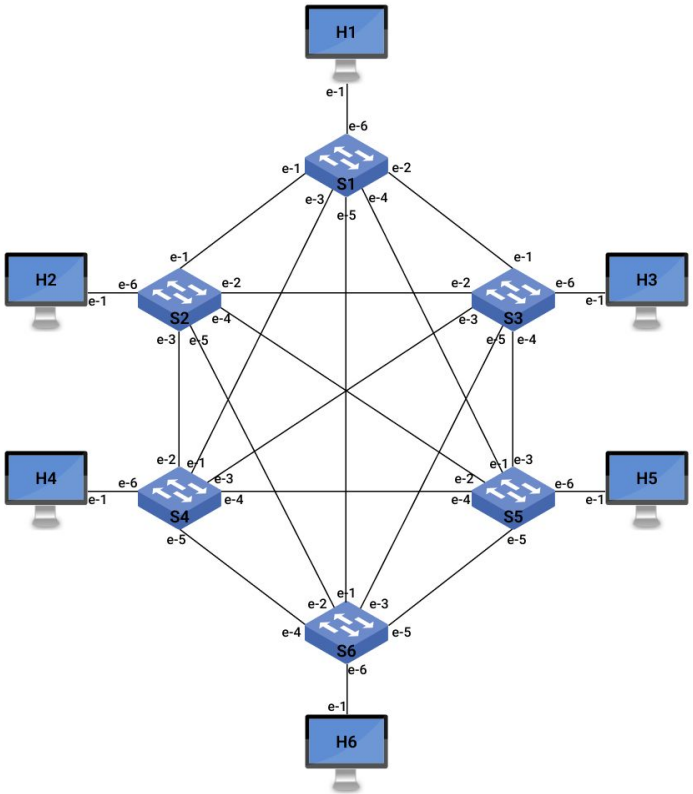
- Implement in ComNetSemu a network slicing mechanism to enable dynamic activation and deactivation of network slices

Users must be able to:

- Activate and deactivate different slices
- Create custom slices and set Quality of Service rules
- Identify the topology, flows and percentage of link capacity for each slice

Default topology

- 6 switches interconnected in a mesh topology
- Each switch is connected to its own host
- No QoS rules applied



Architecture

topology.py

- Setup the default topology and create elements
- Avoid using TCLink because of a collision with OVS QoS rules (explained after)
- Disable IPv6 for each host and switch
- Start the network and perform gratuitous ARP flooding until the STP tree is defined

```
# Generate gratuitous ARP until STP setup is complete
for h in net.hosts:
    h.cmd(f"arping -U -I {h.name}-eth0 $(hostname -I) > /dev/null 2>&1 &")
    h.cmd(f"tcpdump -c 1 'arp' and not host $(hostname -I) > /dev/null 2>&1 && sleep 1 && pkill --nslist net --ns $$ arping > /dev/null 2>&1 &")
    time.sleep(0.1)
```

gui_topology.py

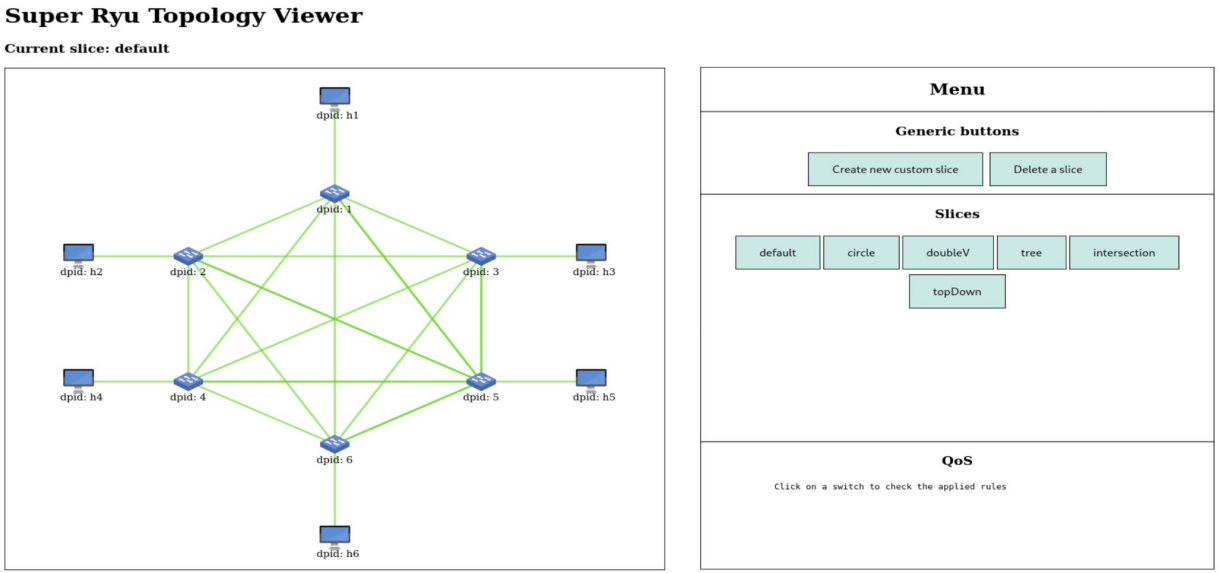
Start the program and import:

- **ofctl_rest**: switch stats
- **rest_topology**: provide API for topology information
- **ws_topology**: websocket to expose topology information
- **rest_conf_switch**: access to OVSDB switches
- **rest_qos_stp**: needed for QoS with STP
- **stplib (default)**: needed for STP functionalities to prevent loops
- **controller**: main Ryu controller

ryu_topology.js

Provide web interface:

- Topology view
- Default slices
- Create and Delete slice
- QoS view



Controller

controller.py

Controls the topology and the slicing mechanism:

- Slice management
- Packet control
- QoS management
- STP
- API

__init__

- Setup STP
- Load template slices
- Register APIs

_packet_in_handler

- Send packet only if communication is allowed by slice configuration
- Check if destination is known, otherwise flooding
- Install flows to avoid flooding every time

restart_stp

- Recalculate STP for each bridge on slice change

parse_active_ports

- Return the active ports for each switch depending on the active slice

```
#Calculate the active ports
def parse_active_ports(self):
    active_ports = {}
    for outer_key, inner_dict in self.sliceToPort["rules"].items():
        for inner_key, value_list in inner_dict.items():
            for value in value_list:
                #Check if switch (outer_key) is already in the array
                if outer_key in active_ports and value not in active_ports[outer_key]:
                    active_ports[outer_key].append(value)
                else:
                    #if not in the array, add the switch in the array with value the port
                    active_ports[outer_key] = [value]
    return active_ports
```

_change_slice

- Remove QoS rules and queues
- Change active slice
- Setup new QoS rules, if any
- Activate and deactivate links based on slice
- Restart STP if needed

APIs

Slice management API endpoints

- **POST: /sliceCreation**: create the slice based on parameter received
- **DELETE: /sliceDeletion/{slicename}**: delete slice based on the slice name
- **GET: /slice/{slicename}**: activate slice based on the slice name
- **GET: /slices**: return the slice list
- **GET: /activeSlice**: return the active slice name

Default API endpoints

- **GET: /v1.0/topology/switches**: get list of switches
- **GET: /v1.0/topology/hosts**: get list of hosts
- **GET: /v1.0/topology/links**: get list of links
- **POST-DELETE: /qos/rules**: set or remove rules
- **POST-DELETE: /qos/queues**: set or remove queues

Slice Management

Slice template

```
"tree": {
        "rules": {
            "1": { "1": [2,3,4,5,6], "2": [1,3,4,5,6], "3": [1,2,4,5,6], "4": [1,2,3,5,6], "5": [1,2,3,4,6], "6": [1,2,3,4,5]},
            "2": { "1": [6], "2": [], "3": [], "4": [], "5": [], "6": [1] },
            "3": { "1": [6], "2": [], "3": [], "4": [], "5": [], "6": [1] },
            "4": { "1": [6], "2": [], "3": [], "4": [], "5": [], "6": [1] },
            "5": { "1": [6], "2": [], "3": [], "4": [], "5": [], "6": [1] },
            "6": { "1": [6], "2": [], "3": [], "4": [], "5": [], "6": [1] }
        },
        "qos": [
            {
                "sw_id": 1,
                "port": "s1-eth6",
                "match": [ { "dst": "10.0.0.1", "src": "10.0.0.2"}, { "dst": "10.0.0.1", "src": "10.0.0.3"} ],
                "queues": [ {"max_rate": "500000"}, {"max_rate": "600000"}, {"max_rate": "800000"}]
            }
        ]
    }
```

A slice is composed by:

- Slice name
- Rules: for each switch and its ports defines on which ports the packet may be forwarded
- QoS: defines which QoS queue should be applied to the defined switches:
  - match: defines the parameter for the QoS rules to be applied
  - queues: defined the queues matching the rules (index off by one, the 0 index queue is the default one)

Change slice

Users are able to change the active slice by using the apposite buttons.

Upon slice change the controller:

- Remove QoS rules and queues
- Change active slice
- Setup new QoS rules, if any
- Activate and deactivate links based on slice
- Restart STP if needed

Delete slice

Users are able to delete a slice by using the apposite button.

Upon slice deletion the controller:

- Remove the slice from the list
- Setup the default slice if the delete one was active

Create slice

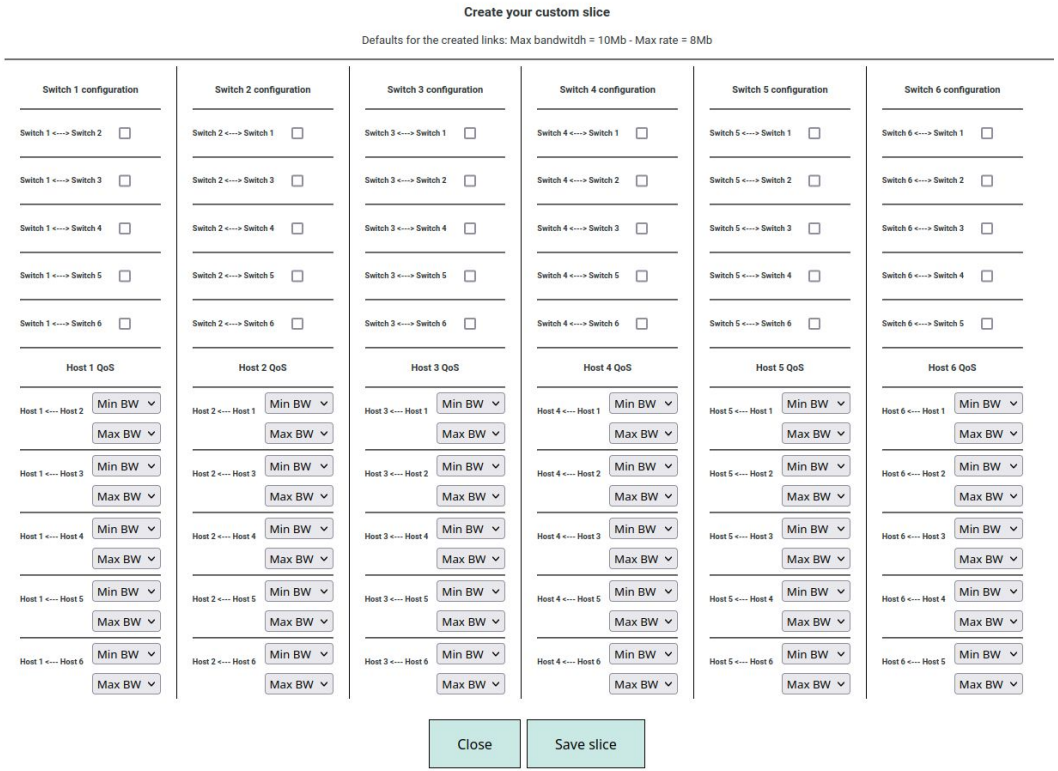
Users are able to create a slice by using the apposite panel.

To create a slice the user must define:

- Slice name
- Active links between switches
- Optionally QoS rules for any host

Upon slice creation the controller:

- Add the slice to the list
- Setup the new slice as the active one by calling change_slice



Errors and Solutions

Errors

- *Problem in the Ryu API*
  - *Restart the ryu service to reload the REST API engine (restoring the default slice)*

Status	Method	Domain	File	Initiator	Type	Transferred	Size
🚫	GET	localhost:8080	switches	fetch		NS_BINDING_ABORT...	
304	GET	localhost:8080	/	document	html	cached	35.02 kB
200	GET	localhost:8080	d3.v3.min.js	script	js	cached	0 B
304	GET	localhost:8080	ryu.topology.js	script	js	cached	25.36 kB
	GET	localhost:8080	switches	ryu.topology.js:370 (fetch)			
200	GET	localhost:8080	slices	ryu.topology.js:426 (fetch)	json	299 B	166 B

Errors

- Problem in the Ryu API
  - *Restart the ryu service to reload the REST API engine (restoring the default slice)*
- After applying a new slice the topology show all the link
  - *Refresh it to observe the (potential) cut of links made by the STP rebuild action.*

Errors

```
[STP][INFO] dpid=0000000000000003: [port=1] Receive superior BPDU.
[STP][INFO] dpid=0000000000000003: [port=6] DESIGNATED_PORT     / BLOCK
hub: uncaught exception: Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/lib/hub.py", line 59, in _launch
    return func(*args, **kwargs)
  File "/usr/lib/python3/dist-packages/ryu/lib/stplib.py", line 540, in recalculate_spanning_tree
    port.down(PORT_STATE_BLOCK, msg_init=init)
  File "/usr/lib/python3/dist-packages/ryu/lib/stplib.py", line 803, in down
    self._change_status(state)
  File "/usr/lib/python3/dist-packages/ryu/lib/stplib.py", line 879, in _change_status
    self.send_bpdu_thread.stop()
  File "/usr/lib/python3/dist-packages/ryu/lib/stplib.py", line 1101, in stop
    hub.joinall([self.thread])
  File "/usr/lib/python3/dist-packages/ryu/lib/hub.py", line 102, in joinall
    t.wait()
AttributeError: 'NoneType' object has no attribute 'wait'
```

- Problem in the Ryu API
  - *Restart the ryu service*
- After applying a new slice
  - *Refresh it to observe t*
- BPDU thread kill failed
  - *override the `hub.py` file or add a check to catch the exception on `bridge.recalculate_spanning_tree()` function call and continues the execution without interruption*

Errors

- Problem in the Ryu API
  - *Restart the ryu service to reload the REST API engine (restoring the default slice)*
- After applying a new slice the topology show all the link
  - *Refresh it to observe the (potential) cut of links made by the STP rebuild action.*
- BPDU thread kill failed
  - *override the `hub.py` file or add a check to catch the exception on `bridge.recalculate_spanning_tree()` function call and continues the execution without interruption*
- Mininet TC(U)link is conflicting with OVS tc usage, so no QoS rule would have be applied:
  - *create the link without any specification of the bandwidth or loss*
  - https://mail.openvswitch.org/pipermail/ovs-discuss/2015-November/019565.html
  - https://github.com/mininet/mininet/issues/243

Errors

- Problem in the Ryu API
  - *Restart the ryu service to reload the REST API engi*
- After applying a new slice the topology show all t
  - *Refresh it to observe the (potential) cut of links ma*
- BPDU thread kill failed
  - *override the `hub.py` file or add a check to catch t `bridge.recalculate_spanning_tree()`*
- Mininet TC(U)link is conflicting with OVS tc usage,
  - *create the link without any specification of the ban*
  - https://mail.openvswitch.org/pipermail/ovs-discuss/2015-November/019565.html
  - https://github.com/mininet/mininet/issues/243
- The standard ryu REST endpoint doesn't recognize dl_type LLDP leading to a *KeyError: 35020 during the call of* http://localhost:8080/qos/rules/0000000000000001
  - *Override of `ryu.app.rest_qos` with `rest_qos_stp.py` adding LLDP type*


```
[
    {
        "switch_id": "0000000000000001",
        "command_result": [
            {
                "qos": [
                    {
                        "qos_id": 0,
                        "priority": 65535,
                        "dl_dst": "01:80:c2:00:00:0e",
                        "dl_type": "lldp",
                        "actions": []
                    },
                    {
                        "qos_id": 0,
                        "priority": 65535,
                        "dl_dst": "01:80:c2:00:00:00",
                        "actions": []
                    },
                    {
```

Install & Run Instructions

Install

- *sudo apt install arping*
- *git clone https://github.com/nicolacarlin/networking-on-demand-slicing-project.git*

Run

Terminal 1:

- *ryu run --observe-links gui_topology.py*

Terminal 2:

- *sudo python3 topology.py*

Thank you for your attention