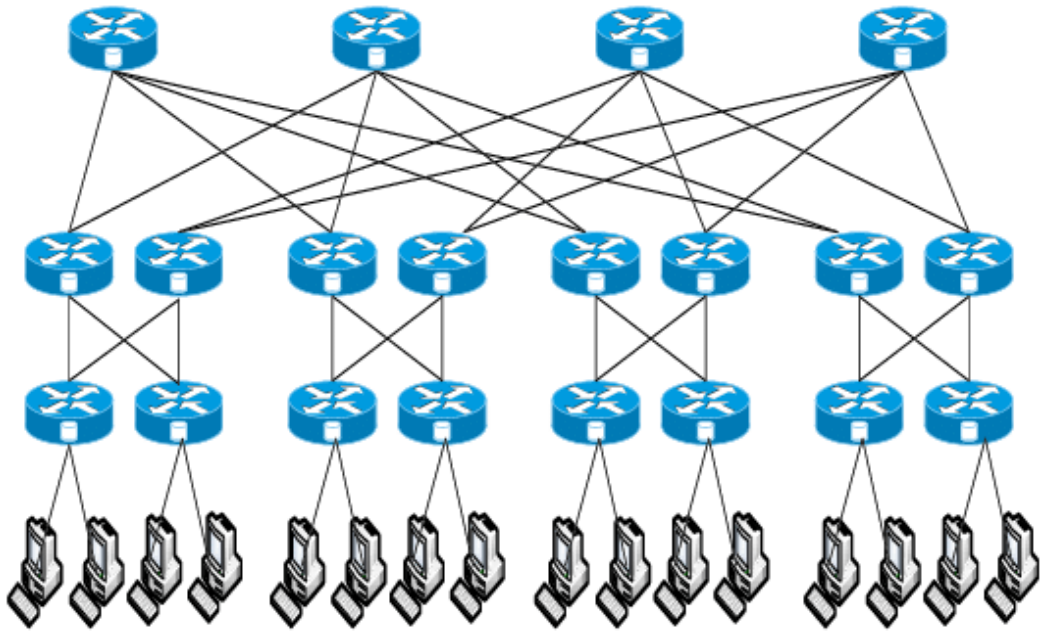
Flow Reservation in Data Centers

ADVANCED NETWORK ARCHITECTURES AND WIRELESS SYSTEMS

FEDERICA PERRONE

VERONICA TORRACA

Requirements



- Hosts are connected through an **SDN-based network**
- The network acts as a **Flow Reservation** system, allowing users to subscribe for a new host-to-host flow, specifying the expected data load (in GiB), through a **REST-based interface**
- The Flow Reservation system guarantees that each physical link is reserved for **at most one** host-to-host **flow**

Functionalities

System **goals**:

- Allow hosts to **request a subscription** for a host-to-host flow with an expected data load (GiB): if no paths are available, the request is denied
- Keep **track** of the current **state of the network**
- **Proactively install flow rules** on the switches part of the subscribed flow path
- Maintain flows until the amount of transmitted data reaches the data load set by the user
- **Automatically deallocate** flows (the sooner, the better)

REST Interface

GET "reserved/links"	Retrieve the list of current reserved links **http://<CONTROLLER_IP>:8080/dc/network/get/reserved/links/json**
GET "reserved/paths"	Retrieve the list of current reserved paths **http://<CONTROLLER_IP>:8080/dc/network/get/reserved/paths/json**
GET "h2h/flow"	Retrieve the list of current reserved host-to-host flow **http://<CONTROLLER_IP>:8080/dc/network/get/h2h/flow/json**
POST "reserve/flow"	Reserve a new host-to-host flow **http://<CONTROLLER_IP>:8080/dc/network/reserve/flow/json**

Flow Reservation

Host-to-host flow subscription

The **Controller** receives subscription request for a new **host-to-host flow**:

- **Check** if **IP addresses** correspond to real hosts in the network
- **Check** if the specified **source** and/or **destination** hosts are not already part of others **subscribed flow** paths
- Retrieve **edge switches** linked to the source and the destination hosts
- **Compute** a configurable number of **potential paths** between source and destination hosts (number set to 20 possible path for increasing the probability to obtain disjoint paths)
- **Iterate** over the list of detected **paths** looking for a path with all **non-reserved links**
- If there exists an **available path**, this is **reserved**: abstraction of the **network state updated** and **flow rules proactively installed** on the switches belonging to that path

Flow rules installations

Source-to-destination path

- **Match**:
  - **ETH_SRC**: Source host MAC address
  - **ETH_DST**: Destination host MAC address
- **Action**:
  - **If** the switch is the **last switch** of the reserved path (an edge switch) **->** send the matching packet through the output port connected to the **destination host**
  - **Else** -> send the matching packet on the port connecting the actual switch with the **next switch** of the reserved path

Destination-to-source path

- **Match**:
  - **ETH_SRC**: Destination host MAC address
  - **ETH_DST**: Source host MAC address
- **Action**:
  - **If** the switch is the **first switch** of the reserved path (an edge switch) **->** send the matching packet through the output port connected to the **source host**
  - **Else** -> send the matching packet on the port connecting the actual switch with the **previous switch** of the reserved path

Hard_timeout and **Idle_timeout** fields are set to **zero** because the flow entries are allocated and deallocated by the SDN-controller in a proactive manner

Traffic monitoring

Controller **monitors traffic** on the **last switch** of the reserved path, to detect when the **flow rules** can be **removed**:

- If possible, obtain **bandwidth consumption** of the **last link** of the path. Otherwise, use a fixed bandwidth
- Use bandwidth to **estimate time** for the **transmission** of the entire data load to the destination host
- When **time elapsed**:
  - Retrieve the **number of bytes delivered** to the destination host
  - If **number of bytes > data load + tolerance threshold** => deallocate host-to-host flow
  - Else => **periodic check** every 15 seconds

Tolerance threshold:

- Evaluating during testing phase
- Set to **0.3%** of data load

Host-to-host flow deallocation

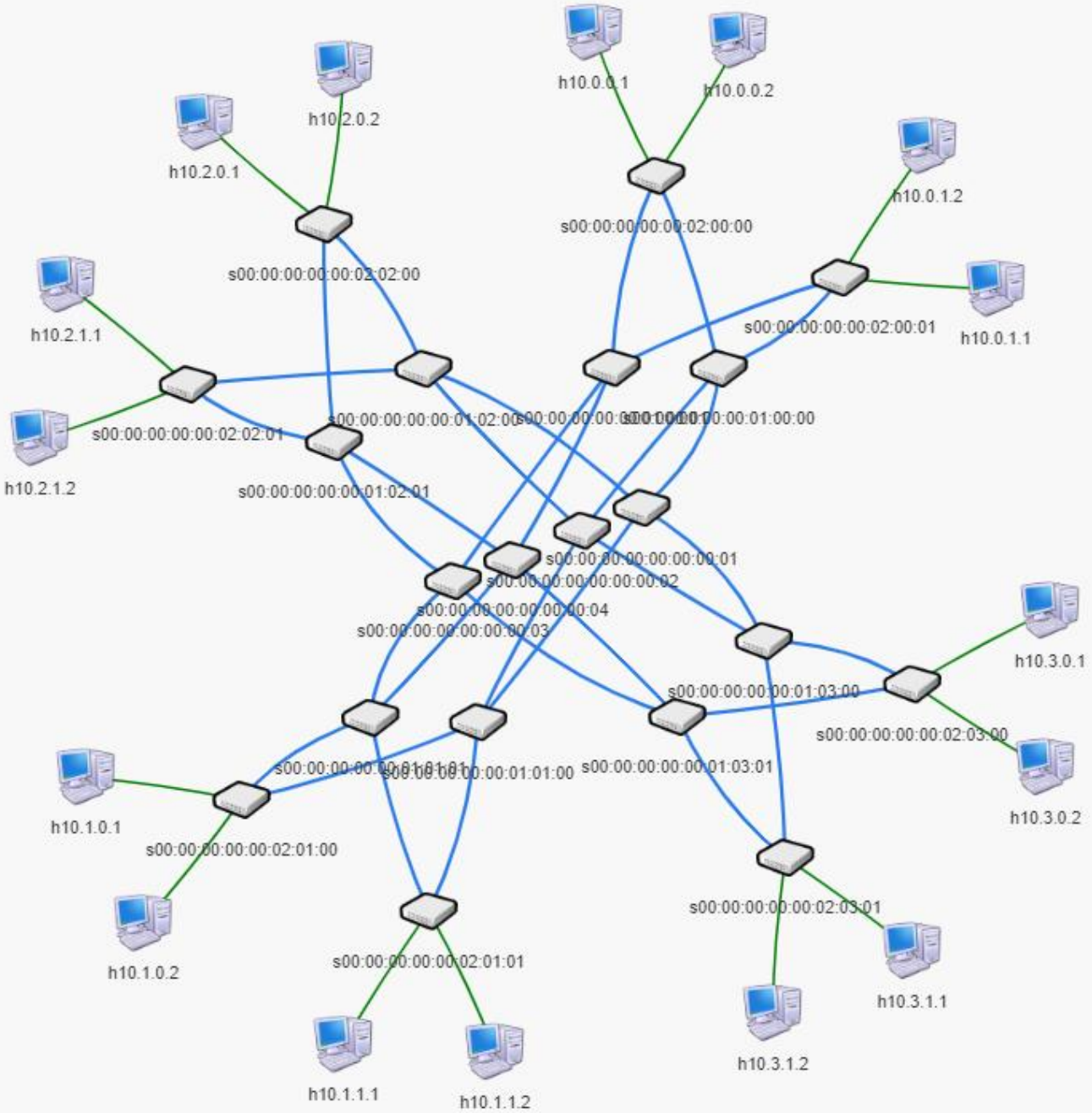
When data load reach destination host, the controller must **deallocate host-to-host flow**

- Send **FlowDelete message** in flow mode to all the switches of the reserved path to **proactively** update their flow table and remove flow rules no longer needed
- **Update network state**

Testing

Testing objectives

- Tests evaluate the system implemented features
- Using **Mininet** emulator a scenario was implemented, to provide a virtual network topology. For tracing the sent packets, **Wireshark** was used to capture the messages
- Inside the virtual network, the available hosts was used to run simple shell scripts to test the nodes communication and message flows, using the developed **Flow Reservation system**



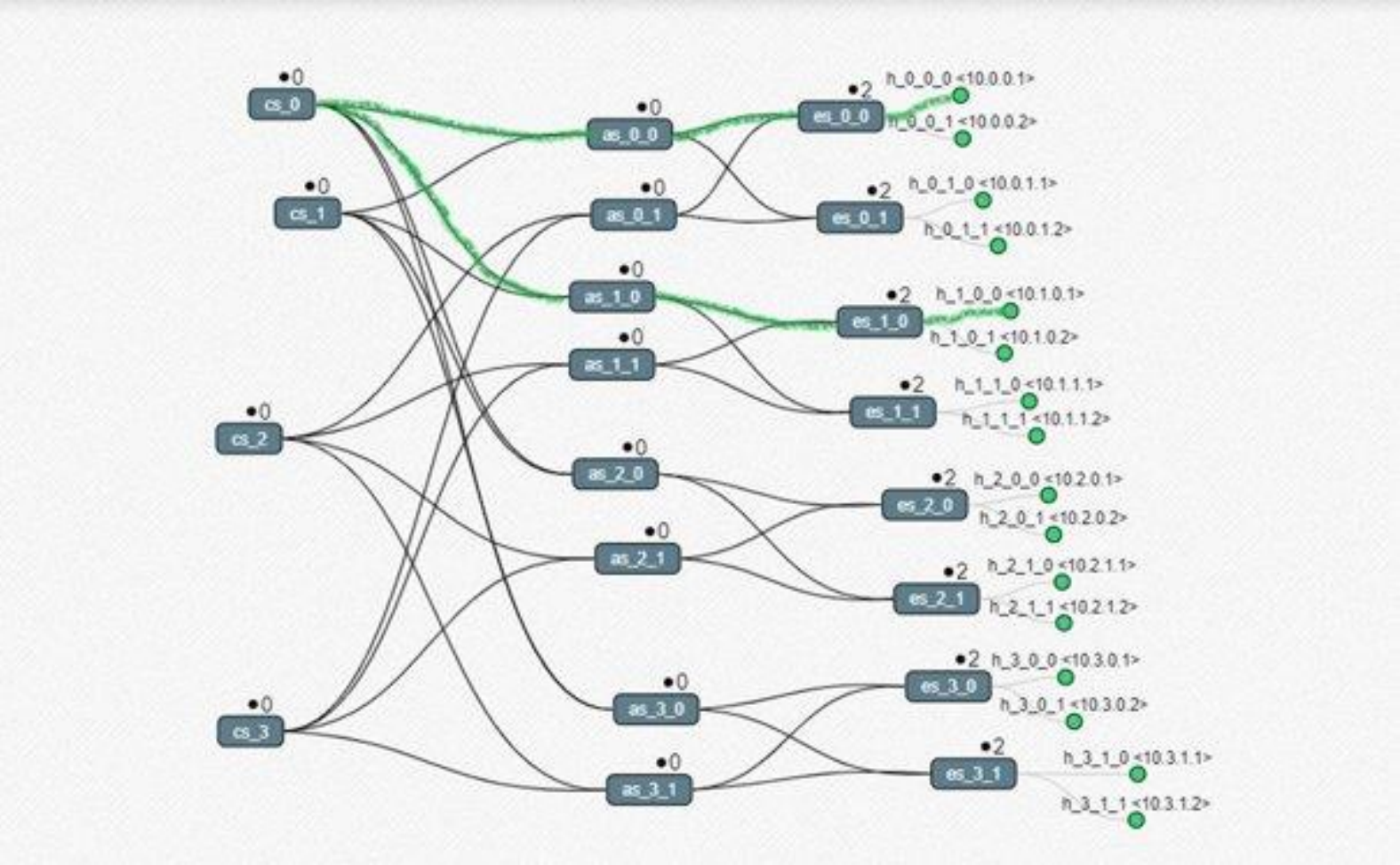
Flow reservation testing

FIRST STEP --> do **POST request** using the `FlowReservation RESTful` interface

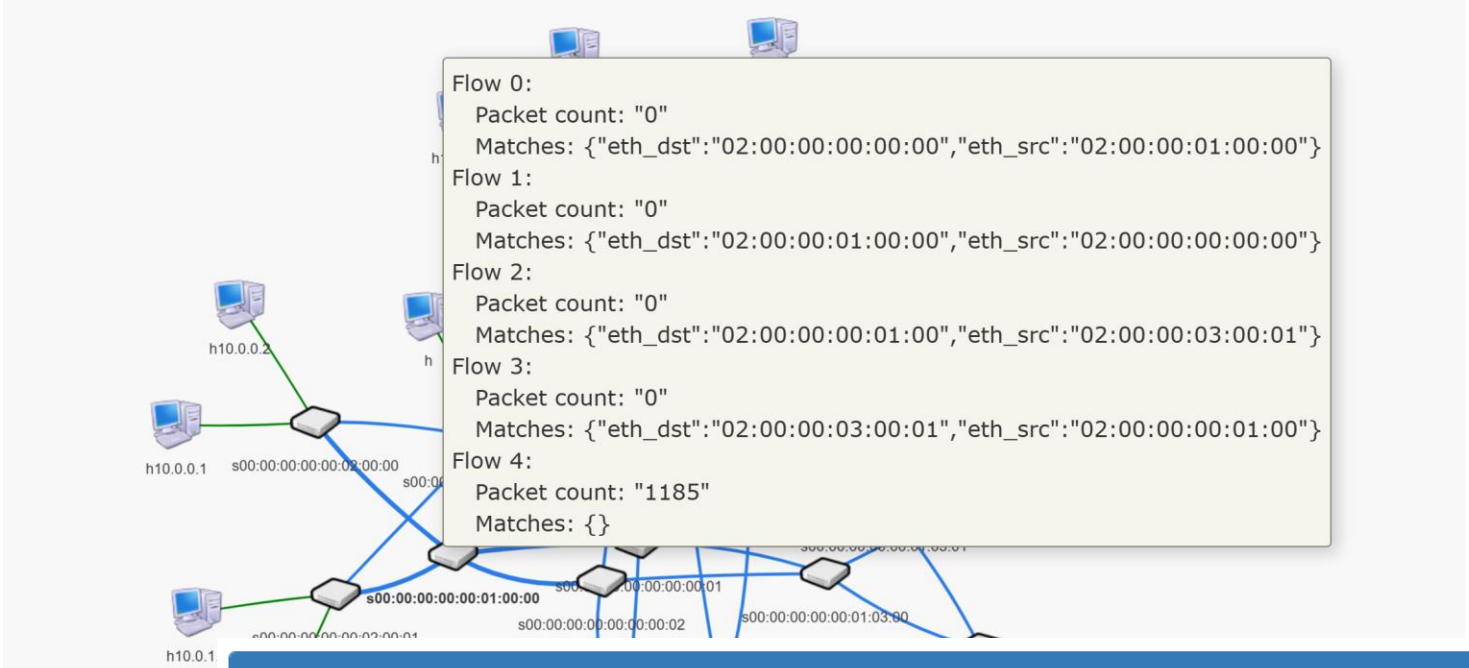
- srcIP: 10.0.0.1, srcMac: 02:00:00:00:00:00
- destIP: 10.1.0.1, destMac: 02:00:00:01:00:00
- dataload: 4GiB

```
student@osboxes:~$ python post_request.py 10.0.0.1 02:00:00:00:00:00 10.1.0.1 02:00:00:01:00:00 4
Namespace(data_load=4.0, dest_ip='10.1.0.1', dest_mac='02:00:00:01:00:00', src_ip='10.0.0.1', src_mac='02:00:00:00:00:00')
OK: path reserved
student@osboxes:~$ python post_request.py 10.0.0.2 02:00:00:00:00:01 10.1.0.2 02:00:00:01:00:01 4
Namespace(data_load=4.0, dest_ip='10.1.0.2', dest_mac='02:00:00:01:00:01', src_ip='10.0.0.2', src_mac='02:00:00:00:00:01')
OK: path reserved
student@osboxes:~$ python post_request.py 10.0.1.1 02:00:00:00:01:00 10.3.0.2 02:00:00:03:00:01 4
Namespace(data_load=4.0, dest_ip='10.3.0.2', dest_mac='02:00:00:03:00:01', src_ip='10.0.1.1', src_mac='02:00:00:00:01:00')
OK: path reserved
student@osboxes:~$ python post_request.py 10.1.1.1 02:00:00:01:01:00 10.3.1.1 02:00:00:03:01:00 4
Namespace(data_load=4.0, dest_ip='10.3.1.1', dest_mac='02:00:00:03:01:00', src_ip='10.1.1.1', src_mac='02:00:00:01:01:00')
OK: path reserved
student@osboxes:~$ python post_request.py 10.0.1.2 02:00:00:00:01:01 10.3.1.2 02:00:00:03:01:01 4
Namespace(data_load=4.0, dest_ip='10.3.1.2', dest_mac='02:00:00:03:01:01', src_ip='10.0.1.2', src_mac='02:00:00:00:01:01')
Reservation requested denied: No path available
```

```
{
  "src_ip: 10.0.0.1": "src_node: 00:00:00:00:00:02:00:00 port: 1 dest_node: 00:00:00:00:00:01:00:00 port: 3; src_node: 00:00:00:00:
00:01:00:00 port: 1 dest_node: 00:00:00:00:00:00:00:01 port: 1; src_node: 00:00:00:00:00:00:00:01 port: 2 dest_node: 00:00:00:00:0
0:01:01:00 port: 1; src_node: 00:00:00:00:00:01:01:00 port: 3 dest_node: 00:00:00:00:00:02:01:00 port: 1; "
}
```

Get **reserved link**
-> do **GET** request to the `FlowReservation RESTful` interface



Get **flow table** of **switches** belonging to the **reserved path** to check if flow rules was installed correctly

Flow Table

Show 10 entries Search:

Table No	Pkt.Count	Byte	Duration(s)	Priority	IdleTimeoutSec	HardTimeoutSec	Flags	Instructions
0x0	22284	526050600	67	32768	0	0		output=1
0x0	19767	1306834	67	32768	0	0		output=3
0x0	279	18974	286	0	0	0		output=controller

SECOND STEP --> **send file** from the source host to the destination host using **netcat** and check if the file packets were correctly delivered to the destination with **Wireshark**

No.	Time	Source	Destination	Protocol	Length	Info
5	22.233774216	10.0.0.1	10.1.0.1	TCP	74	45880 → 1234 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=299623401 TSecr=0 WS=512
6	22.238782595	10.1.0.1	10.0.0.1	TCP	74	1234 → 45880 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=341267451 TSecr=299623401 WS=512
7	22.238801754	10.0.0.1	10.1.0.1	TCP	66	45880 → 1234 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=299623406 TSecr=341267451
8	22.239046793	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=1 Ack=1 Win=29696 Len=4344 TSval=299623406 TSecr=341267451
9	22.239076628	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=4345 Ack=1 Win=29696 Len=4344 TSval=299623406 TSecr=341267451
10	22.242465419	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=8689 Ack=1 Win=29696 Len=4344 TSval=299623406 TSecr=341267451
11	22.242788683	10.0.0.1	10.1.0.1	TCP	1514	45880 → 1234 [ACK] Seq=13033 Ack=1 Win=29696 Len=1448 TSval=299623410 TSecr=341267451
12	22.253600577	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=4345 Win=37888 Len=0 TSval=341267456 TSecr=299623406
13	22.253621924	10.0.0.1	10.1.0.1	TCP	5858	45880 → 1234 [ACK] Seq=14481 Ack=1 Win=29696 Len=5792 TSval=299623420 TSecr=341267456
14	22.253809372	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=8689 Win=46592 Len=0 TSval=341267460 TSecr=299623406
15	22.253824745	10.0.0.1	10.1.0.1	TCP	5858	45880 → 1234 [ACK] Seq=20273 Ack=1 Win=29696 Len=5792 TSval=299623421 TSecr=341267460
16	22.254188310	10.0.0.1	10.1.0.1	TCP	2962	45880 → 1234 [ACK] Seq=26065 Ack=1 Win=29696 Len=2896 TSval=299623421 TSecr=341267460
17	22.254220567	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=13033 Win=55296 Len=0 TSval=341267460 TSecr=299623406
18	22.254229565	10.0.0.1	10.1.0.1	TCP	5858	45880 → 1234 [ACK] Seq=28961 Ack=1 Win=29696 Len=5792 TSval=299623421 TSecr=341267460
19	22.254591152	10.0.0.1	10.1.0.1	TCP	5858	45880 → 1234 [ACK] Seq=34753 Ack=1 Win=29696 Len=5792 TSval=299623421 TSecr=341267460
20	22.255413236	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=14481 Win=58368 Len=0 TSval=341267460 TSecr=299623410
21	22.255426431	10.0.0.1	10.1.0.1	TCP	2962	45880 → 1234 [ACK] Seq=40545 Ack=1 Win=29696 Len=2896 TSval=299623422 TSecr=341267460
22	22.255565764	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=20273 Win=69632 Len=0 TSval=341267470 TSecr=299623420
23	22.255576293	10.0.0.1	10.1.0.1	TCP	2962	45880 → 1234 [ACK] Seq=43441 Ack=1 Win=29696 Len=2896 TSval=299623422 TSecr=341267470
24	22.255703629	10.1.0.1	10.0.0.1	TCP	66	1234 → 45880 [ACK] Seq=1 Ack=46337 Win=121856 Len=0 TSval=341267473 TSecr=299623422
25	22.255711567	10.0.0.1	10.1.0.1	TCP	2962	45880 → 1234 [ACK] Seq=46337 Ack=1 Win=29696 Len=2896 TSval=299623422 TSecr=341267470
26	22.255837427	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=49233 Ack=1 Win=29696 Len=4344 TSval=299623423 TSecr=341267473
27	22.256417108	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=53577 Ack=1 Win=29696 Len=4344 TSval=299623423 TSecr=341267473
28	22.256473291	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=57921 Ack=1 Win=29696 Len=4344 TSval=299623423 TSecr=341267473
29	22.256577949	10.0.0.1	10.1.0.1	TCP	4410	45880 → 1234 [ACK] Seq=62265 Ack=1 Win=29696 Len=4344 TSval=299623423 TSecr=341267473

THIRD STEP --> **periodically check** if the data load reaches the destination host, finally **deallocate host-to-host flow**

```
2023-03-27 12:16:53.204 INFO  [n.f.u.f.FlowReservation] Enable statistics: wait collection of statistics
2023-03-27 12:16:53.403 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-03-27 12:16:53.435 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-03-27 12:17:04.304 INFO  [n.f.u.f.FlowReservation] bytes delivered: 2497097524
2023-03-27 12:17:04.304 INFO  [n.f.u.f.FlowReservation] packets: 68614
2023-03-27 12:17:04.304 INFO  [n.f.u.f.FlowReservation] File not delivered to destination yet. Wait until next check
2023-03-27 12:17:08.886 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-03-27 12:17:08.909 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-03-27 12:17:09.616 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-03-27 12:17:19.304 INFO  [n.f.u.f.FlowReservation] Enable statistics: wait collection of statistics
2023-03-27 12:17:23.942 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-03-27 12:17:24.467 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-03-27 12:17:30.406 INFO  [n.f.u.f.FlowReservation] bytes delivered: 3365766472
2023-03-27 12:17:30.406 INFO  [n.f.u.f.FlowReservation] packets: 93308
2023-03-27 12:17:30.406 INFO  [n.f.u.f.FlowReservation] File not delivered to destination yet. Wait until next check
2023-03-27 12:17:31.166 ERROR [n.f.c.w.SwitchStatisticsResource] Invalid or unimplemented stat request type features
2023-03-27 12:17:39.74 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-03-27 12:17:45.412 INFO  [n.f.u.f.FlowReservation] Enable statistics: wait collection of statistics
2023-03-27 12:17:54.102 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-03-27 12:17:56.512 INFO  [n.f.u.f.FlowReservation] bytes delivered: 4305592110
2023-03-27 12:17:56.512 INFO  [n.f.u.f.FlowReservation] packets: 120263
2023-03-27 12:17:56.512 INFO  [n.f.u.f.FlowReservation] File delivered to destination
2023-03-27 12:17:56.512 INFO  [n.f.u.f.FlowReservation] Deallocate h2h flow
2023-03-27 12:17:56.512 INFO  [n.f.u.f.FlowReservation] Data structure updated correctly
2023-03-27 12:17:56.516 INFO  [n.f.u.f.FlowReservation] Flow rules removed correctly
```

Troubleshooting

Host-to-host **flow deallocation**

- Sometimes, flow are **not deallocated** because some other modules **disable statistics** collection
- Bandwidth consumption is recalculated from scratch each time, and this cause a very **large delay** in flow deallocation

Tolerance threshold

- **Estimating** during test phase
- Different values are obtained, from a minimum value of about 0.2% (more likely) to a maximum value of 0.5%
- Considered the experiments results, the threshold was set to **0.3%** in order to deallocate most paths, because the choice was between the probability of never deallocating a path and the probability of deallocating some of them before all data was transmitted
- To manage the probability of never deallocating a path, a counter was introduced

Conclusions

- The **flow rules** are **correctly installed** on the switch of the chosen path, in a **proactive** manner
- After an estimated time, the controller check if the entire data load reached the destination, in that case the **flow rules** are **removed** from the switches, otherwise check again periodically
- A **tolerance threshold** for the data load was considered
- This approach works well in most cases, but sometimes even for the statistics collection disabled (out of our control) or for a too small tolerance threshold (random cases) the **flow deallocation fails**

Quantum Internet

```
federica@DESKTOP-7D3FIU6:~/quantum/progetto$ /bin/python3 /home/federica/quantum/progetto/main.py
[0.0] Repeater 1: Starting MS Protocol instance
[0.0] Repeater 1: Starting EPS
[0.0] Repeater 2: Starting MS Protocol instance
[0.0] Repeater 2: Waiting for START message
[75249.0] Repeater 1: Sending START message with value 225259
[225249.0] Repeater 2: Received START message
[225259.0] Repeater 2: Starting entanglement generation
[225259.0] Repeater 1: Starting entanglement generation
[225259.0] Repeater 1: Latched photon at attempt 0
[225259.0] Repeater 2: Latched photon at attempt 0
[376819.0] Repeater 1: Entanglement generation successful at attempt 0
[376819.0] Repeater 2: Entanglement generation successful at attempt 0
[376819.0] Repeater 2: Starting MS Protocol instance
[376819.0] Repeater 2: Waiting for START message
[376819.0] Repeater 1: Starting MS Protocol instance
[376819.0] Repeater 1: Starting EPS
[376846.0] Repeater 1: Sending START message with value 526859
[526846.0] Repeater 2: Received START message
[526859.0] Repeater 2: Starting entanglement generation
[526859.0] Repeater 1: Starting entanglement generation
[526880.0] Repeater 1: Latched photon at attempt 2
[526880.0] Repeater 2: Latched photon at attempt 2
[678419.0] Repeater 2: Entanglement generation successful at attempt 2
[678419.0] Repeater 1: Entanglement generation successful at attempt 2
[678419.0] Repeater 1: Starting MS Protocol instance
[678419.0] Repeater 1: Starting EPS
[678419.0] Repeater 2: Starting MS Protocol instance
[678419.0] Repeater 2: Waiting for START message
[678427.0] Repeater 1: Sending START message with value 828439
[828427.0] Repeater 2: Received START message
[828439.0] Repeater 2: Starting entanglement generation
[828439.0] Repeater 1: Starting entanglement generation
[828454.0] Repeater 1: Latched photon at attempt 1
[828461.0] Repeater 2: Latched photon at attempt 2
[979999.0] Repeater 1: Entanglement generation failed. Starting new round
[979999.0] Repeater 2: Entanglement generation failed. Starting new round
[980000.0] Repeater 1: Latched photon at attempt 0
[980000.0] Repeater 2: Latched photon at attempt 0
[1131559.0] Repeater 1: Entanglement generation successful at attempt 0
[1131559.0] Repeater 2: Entanglement generation successful at attempt 0
[1131559.0] Repeater 2: Starting MS Protocol instance
[1131559.0] Repeater 2: Waiting for START message
[1131559.0] Repeater 1: Starting MS Protocol instance
[1131559.0] Repeater 1: Starting EPS
[1131561.0] Repeater 1: Sending START message with value 1281579
[1281561.0] Repeater 2: Received START message
[1281579.0] Repeater 2: Starting entanglement generation
[1281579.0] Repeater 1: Starting entanglement generation
[1281589.0] Repeater 1: Latched photon at attempt 1
[1281589.0] Repeater 2: Latched photon at attempt 1
[1433139.0] Repeater 1: Entanglement generation successful at attempt 1
[1433139.0] Repeater 2: Entanglement generation successful at attempt 1
[1433139.0] Repeater 2: Four qubits are entangled with fidelity F0(qubit0)=0.8574999999999998, F0(qubit1)=0.8574999999999998, F0(qubit2)=0.8574999999999998 and F0(qubit3)=0.8574999999999998
[1433139.0] Repeater 2: Start the first step of purification (on qubit0 and qubit1)
[1433139.0] Repeater 1: Four qubits are entangled with fidelity F0(qubit0)=0.8574999999999998, F0(qubit1)=0.8574999999999998, F0(qubit2)=0.8574999999999998 and F0(qubit3)=0.8574999999999998
[1433139.0] Repeater 1: Start the first step of purification (on qubit0 and qubit1)
[1583141.0] Purification successful
[1583141.0] Fidelity of the new qubit pair with respect to the Bell state F1(qubit0)=0.8907221786124025
[1583141.0] Repeater 2: Continue the first step of purification (on qubit2 and qubit3)
[1583141.0] Purification successful
[1583141.0] Fidelity of the new qubit pair with respect to the Bell state F1(qubit0)=0.8907221786124025
[1583141.0] Repeater 1: Continue the first step of purification (on qubit2 and qubit3)
[1733143.0] Purification successful
[1733143.0] Fidelity of the new qubit pair with respect to the Bell state F1(qubit2)=0.8907221786124025
[1733143.0] Repeater 2: Start the second step of purification (on qubit0 and qubit2)
[1733143.0] Purification successful
[1733143.0] Fidelity of the new qubit pair with respect to the Bell state F1(qubit2)=0.8907221786124025
[1733143.0] Repeater 1: Start the second step of purification (on qubit0 and qubit2)
[1883145.0] Purification successful
[1883145.0] Fidelity of the new qubit pair with respect to the Bell state F2(qubit0)=0.8207604768377772
[1883145.0] Purification successful
[1883145.0] Fidelity of the new qubit pair with respect to the Bell state F2(qubit0)=0.8207604768377772
```