REOC PROJECT

2024-2025

Noël Jumin
Clément Gauché

Noël Jumin
Clément Gauché

Introduction

In the context of the Master REOC and following the various courses on networks utilizing SDN and VNFs, we conducted a network simulation to address the challenges of saturation caused by traffic in a specific zone. This simulation allowed us to implement a proof of concept that leverages monitoring VNFs and traffic redirection through using SDN rules. Our objective was to demonstrate that these methods effectively prevent network limitations while optimizing communication between zones and maintaining overall network performance.

Problem Statement

The issue we aim to address is network saturation caused by a specific zone. To tackle this, we will rely on the following network plan, which will be detailed later:

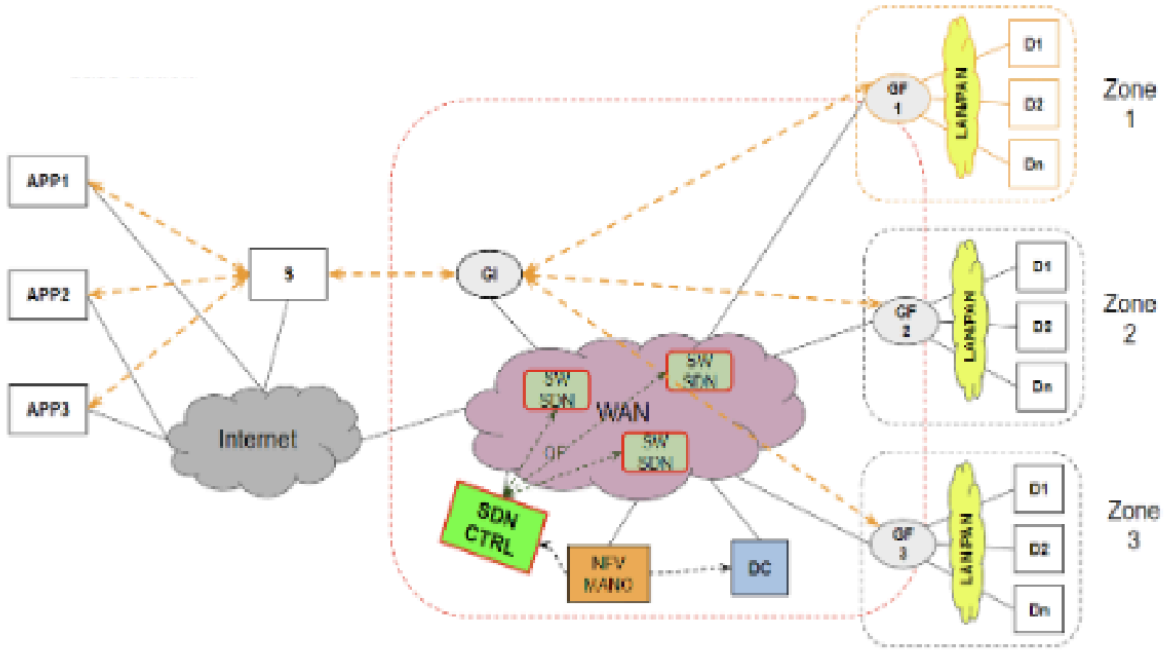

Figure 1. Application network configuration

To take action, we must first detect when a zone exceeds a certain threshold and then implement measures to reduce traffic and prevent network saturation.

Possible Approaches

Several monitoring options are available:

- **Application response time for the GI**: Measuring latency using ping.
- **Resource status (health monitoring)**: Verifying the health of network components.
- **Application-level monitoring through a proxy VNF**: For example, monitoring a VNF handling caching.
- **Monitoring application throughput**: Measuring the number of requests per second or bytes per second for SDN (explained in the next point).
- **OpenFlow SDN rules**: Adding a rule in an SDN switch to increment a counter with every new message on the port of a given zone. This allows us to estimate the throughput of each zone.

- **Application-layer sensor modification**: Calculating the time between sending a message and receiving a response (similar to an application-level "ping").

In our case, we decided to focus on application-level monitoring by deploying a VNF that acts as an intermediary between the zones and the intermediate gateway to count the frames passing through.

Actions Considered

- **Deployment of a second intermediate gateway with traffic redirection/sharing:** Redirect traffic from Zone 1 to a new intermediate gateway, while maintaining traffic from Zones 2 and 3 through the same original gateway.
- **Scheduler or Load Balancer deployment:** Distribute the load from Zones 1, 2, and 3 evenly. In cases of overload, certain frames may be rejected to balance the network effectively.
- **Layer 7 VNF deployment for caching or rate limiting:** Deploy a Layer 7 Virtual Network Function between a zone and the intermediate gateway to cache data and reduce network pollution while waiting for traffic throughput to decrease.
- **Traffic rejection or rate limiting at Layer 3:** Use SDN rules to reject part of the frames, for example, to optimize network traffic and maintain stability.

We have opted to focus on deployment of a second intermediate gateway, redirecting traffic from Zone 1 to a newly created gateway VNF specifically for this purpose.

Noël Jumin
Clément Gauché

Network Topology Implementation

Below, you can see the topology we deploy with Mininet:

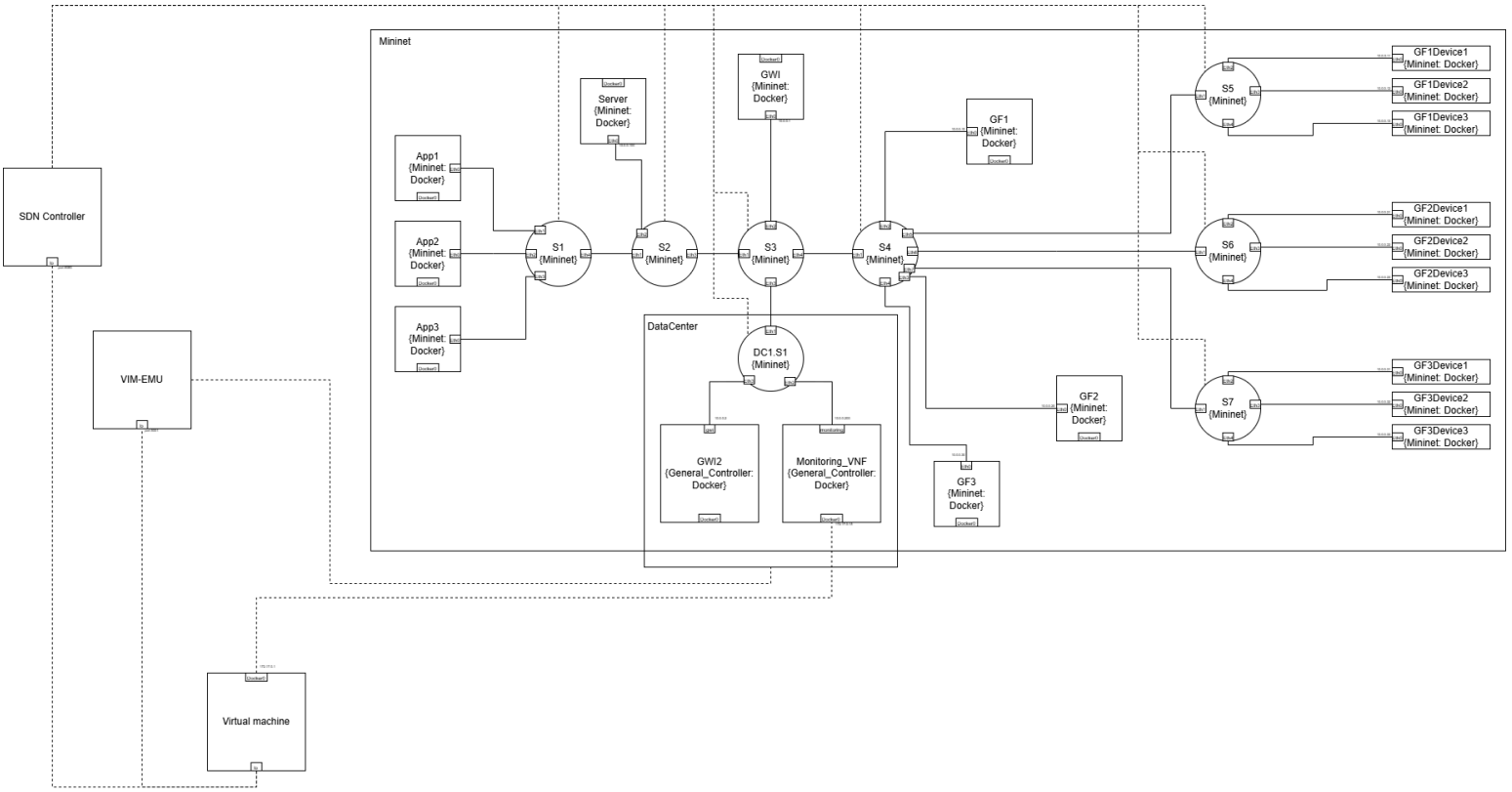

Figure 2. Topology created to simulate our network

Each machine is connected to the network through an SDN switch, which is itself connected to the SDN controller. These SDN switches can be accessed via a REST API to:

- Monitor a specific switch, for example, to determine the throughput passing through it.
- Add rules to modify fields in specific frames.

For our application, we have extensively used the latter functionality.

This topology also includes a data center that hosts our VNFs. By default, deploying a new device on a network purely through software is quite challenging. Thanks to the data center, this becomes possible because it contains a reprogrammable SDN switch that automatically creates a new interface when a new VNF is added, thereby connecting it to the rest of the network.

These VNFs, like all other containers, are equipped with a Docker interface, allowing them to be addressed from outside the Mininet network. In this case, this is possible

from our machine, as it also has a Docker interface. For instance, this makes it possible to access a specific service on one of the Dockers via an HTTP command or to open a terminal inside the container.

To add a VNF, we use the VIM, which also has a REST API that can be accessed via a specific port.

It is important to note that VNFs do not need to exist when the Mininet network is created. However, the data center must be created simultaneously with the rest of the network and connected to it, even if it starts empty.

Each block within the Mininet network, except the data center and switches, represents a container we defined before starting the topology. We provide these Docker containers with a program to run at launch, depending on their intended tasks. Additionally, we can assign them specific IP and MAC addresses, making it easy to configure SDN rules, which will be discussed later.

Our application is divided into two VNFs:

1. The first VNF monitors the network and collects information about it. In our case, this VNF retrieves the number of requests sent per second by each final gateway.
2. The second VNF simulates an additional intermediate gateway to optimize the network by redirecting one of the data streams to it.

Development of the Monitoring VNF

To implement the monitoring VNF, we began by replicating the code of the intermediate gateway. The reason is that to intercept every request, the monitoring VNF will need to host the same services as the intermediate gateway, particularly the same paths.
Once a request is received, the monitoring VNF increments the counter for the corresponding address and forwards the request to the intermediate gateway.
However, the process is not that straightforward. We will break it into several stages:

SDN Redirection

When the VNF is created, two rules are added to the S3 switch connected to the data center:

Redirect Frames from Final Gateways to the Monitoring VNF

As the name suggests, this rule redirects all frames originating from the final gateways to the monitoring VNF.

- We match all frames from switch S4 destined for the intermediate gateway.
- For matched frames, the following fields are updated:
    - **Destination IP Address**
    - **Destination MAC Address**
- Both fields are replaced with those of the monitoring VNF. The modified frames are then sent back to the switch DC1.S1 connected to the VNFs.

Redirect Frames from the Monitoring VNF to Final Gateways

Since the final gateways use HTTP requests, they rely on TCP. Establishing a TCP connection involves a three-way handshake before transmitting data. Without this rule, the final gateways cannot recognize the monitoring VNF as the responder to their SYN messages sent to the intermediate gateway. Thus, "spoofing" is necessary.

- This second rule matches all frames destined for the final gateways at the exit of switch DC1.S1. The following fields are modified:
    - **Source IP Address**

    - **Source MAC Address**
- Both fields are replaced with those of the intermediate gateway. The altered frames are then sent back to switch S4, where the final gateways are connected.

Internal Functioning of the Monitoring VNF

As noted earlier, the VNF's operation is straightforward.

- It receives requests from the final gateways, increments a counter for each gateway, and every 5 seconds, calculates the number of requests received per second for each final gateway.
- A service is also added to enable retrieving the calculated throughput from outside the Mininet network via HTTP. For instance:

```
sdci@sdci-vm:~$ curl -X get 172.17.0.19:8181/bitrate
{"10.0.0.10":0.9996459307683112,"10.0.0.20":0.9996459307683112,"10.0.0.30
":0.9996459307683112}
```

Adjusting the TOS Field

Before communicating with the intermediate gateway and forwarding requests, the monitoring VNF modifies the socket to update the Type of Service (TOS) parameter. This adjustment identifies the final gateway that originated the request.

The tcpdump capture on the intermediate gateway illustrates the following:

- The three phases of the TCP protocol:
    - **Connection initiation** (blue)
    - **Data exchange**
    - **Disconnection** (orange)
- The modified TOS field, set to 0x8. Based on the enumeration defined in the code, this value corresponds to Gateway Final 2 as the source of the request.

Here is a sample capture:

```
10:33:41.436812 IP (tos 0x8, ttl 64, id 51779, offset 0, flags [DF],
proto TCP (6), length 60)
    10.0.0.200.56154 > 10.0.0.1.8181: Flags [S], cksum 0x14f7 (incorrect
-> 0x4e10), seq 411797328, win 42340, options [mss 1460,sackOK,TS val
3272925568 ecr 0,nop,wscale 9], length 0
...
10:33:41.437410 IP (tos 0x8, ttl 64, id 51781, offset 0, flags [DF],
proto TCP (6), length 214)
    10.0.0.200.56154 > 10.0.0.1.8181: Flags [P.], cksum 0x1591 (incorrect
-> 0xae5e), seq 1:163, ack 1, win 83, options [nop,nop,TS val 3272925568
ecr 1487634475], length 162
...
{"Name":"device3_gf2","Data":1742,"CreationTime":1735554821427,"Reception
Time":null}
10:33:41.441824 IP (tos 0x0, ttl 64, id 54687, offset 0, flags [DF],
proto TCP (6), length 291)
    10.0.0.100.8080 > 10.0.0.1.37330: Flags [P.], cksum 0x157a (incorrect
-> 0x9b37), seq 961:1200, ack 1255, win 1118, options [nop,nop,TS val
1844172574 ecr 142603713], length 239: HTTP, length: 239
        HTTP/1.1 201 Created
        X-Powered-By: Express
        Content-Type: text/plain; charset=utf-8
        Content-Length: 7
        ETag: W/"7-rM9AyJuqT6iOan/xHh+AW+7K/T8"
        Date: Mon, 30 Dec 2024 10:33:41 GMT
        Connection: keep-alive
        Keep-Alive: timeout=5

        Created
...
10:33:41.443243 IP (tos 0x8, ttl 64, id 51784, offset 0, flags [DF],
proto TCP (6), length 52)
    10.0.0.200.56154 > 10.0.0.1.8181: Flags [F.], cksum 0x14ef (incorrect
-> 0x7b9a), seq 254, ack 230, win 83, options [nop,nop,TS val 3272925574
...
    10.0.0.200.56154 > 10.0.0.1.8181: Flags [.], cksum 0x14ef (incorrect
-> 0x7b99), ack 231, win 83, options [nop,nop,TS val 3272925574 ecr
1487634481], length 0
```

This demonstrates the altered TOS field, providing information about the originating final gateway while maintaining the standard TCP process flow.

Additional Observations

We made the deliberate decision not to add rules for modifying the source IP address of the messages and left the monitoring VNF as the source with its IP address 10.0.0.200.

This decision was based on the fact that, in this particular case, the source address is not utilized by the intermediate gateway. Therefore, keeping the monitoring VNF's IP as the source simplifies the configuration and reduces the number of rules required on the switch. This optimization contributes to lowering the processing overhead on the switch.

Development of the Intermediate Gateway VNF

To implement the new intermediate gateway VNF, we started by duplicating the code used for the original intermediate gateway. This ensured that the new gateway maintained equivalent functionality while enabling us to offload part of the traffic. Compared to the monitoring VNF, this development involved simpler modifications since all changes were confined to the SDN layer.

SDN Redirection

Upon creating the new VNF, we added two rules to the DC1.S1 switch, the internal datacenter switch connected to both the VNFs and the external switch S3:

Redirect Frames from Monitoring VNF to the New Intermediate Gateway

This rule ensures all frames from a specific zone originating from the monitoring VNF and destined for the original intermediate gateway are redirected to the new gateway. We match all frames that are from the monitoring VNF. Targeting the intermediate gateway with a specific TOS value (for example 0x4 which is the TOS for gateway final 1).

- For matched frames, the following fields are updated:
    - **Destination IP Address**
    - **Destination MAC Address**
- Both fields are replaced with those of the new gateway.

These modified frames are then sent to the new intermediate gateway.

Redirect Frames from New Intermediate Gateway to the Monitoring VNF

Since the final gateways utilize HTTP over TCP, proper handling of the TCP three-way handshake is crucial to ensure the monitoring VNF recognizes responses from the new gateway. This rule “spoofs” responses to align with the original gateway's identity.

- Frames exiting DC1.S1 and destined for the monitoring VNF are matched, and the following fields are modified:
    - **Source IP Address**
    - **Source MAC Address**
- Both fields are replaced with those of the original intermediate gateway.

These altered frames are then sent to the monitoring VNF.

The service logs confirm proper functionality in a scenario where Zone 1 (gateway final 1) is redirected to our new gateway.

- **Intermediate Gateway Logs:**

```
{
  Name: 'device1_gf3',
  Data: 1545,
  CreationTime: 1735554232060,
  ReceptionTime: null
}
Created
{
  Name: 'device3_gf2',
  Data: 1546,
  CreationTime: 1735554233031,
  ReceptionTime: null
}
Created
{
  Name: 'device2_gf3',
  Data: 1545,
  CreationTime: 1735554233119,
  ReceptionTime: null
}
Created
```

Only devices from final gateways 2 and 3 are processed, as expected.

- **Intermediate Gateway VNF Logs:**

```
{
  Name: 'device1_gf1',
  Data: 2737,
  CreationTime: 1735560077717,
  ReceptionTime: null
}
Created
{
  Name: 'device2_gf1',
  Data: 2737,
  CreationTime: 1735560078427,
  ReceptionTime: null
```

```
}
Created
{
  Name: 'device3_gf1',
  Data: 2737,
  CreationTime: 1735560080590,
  ReceptionTime: null
}
Created
{
  Name: 'device1_gf1',
  Data: 2738,
  CreationTime: 1735560080721,
  ReceptionTime: null
}
Created
```

Requests from devices connected to final gateway 1 are successfully processed by our new intermediate gateway, with no data loss.

- **Service Logs:**

```
{
  Name: 'device1_gf1',
  Data: 2764,
  CreationTime: 1735560158771,
  ReceptionTime: null
}
{
  Name: 'device1_gf3',
  Data: 2761,
  CreationTime: 1735560158773,
  ReceptionTime: null
}
{
  Name: 'device2_gf1',
  Data: 2764,
  CreationTime: 1735560159485,
  ReceptionTime: null
}
{
  Name: 'device3_gf2',
  Data: 2762,
  CreationTime: 1735560159478,
  ReceptionTime: null
```

```
}
{
  Name: 'device1_gf2',
  Data: 2763,
  CreationTime: 1735560159483,
  ReceptionTime: null
}
```

The service layer log demonstrates successful reception of data from all final gateways. The system operates seamlessly, receiving and processing data from all final gateways without any issues.

GUI

We chose to integrate a user interface to make the system more user-friendly. The interface is presented as a page with buttons and a text area to display the feedback from commands sent.

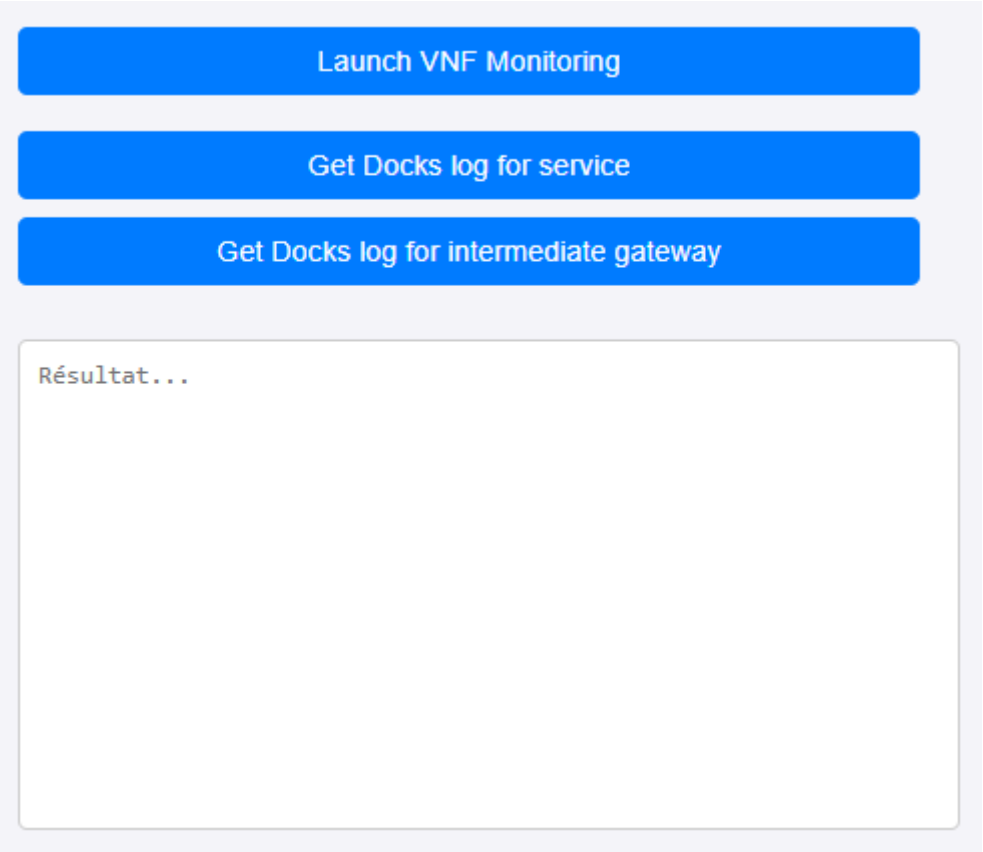

Figure 3. User interface

Capabilities of the Interface:

- **Starting VNFs**: The UI allows users to start the different VNFs in the correct sequence. Since certain VNFs depend on the initialization of others, the UI guides users to launch them in a specified order.
- **Log Retrieval from Docker Containers**: The interface can fetch logs from different Docker containers, providing real-time data for monitoring and troubleshooting.
- **Monitoring VNF Throughput**: The UI enables users to retrieve the throughput of the monitoring VNF, providing a crucial metric for understanding network performance.
- **Traffic Redirection Configuration**: Before launching the intermediate gateway VNF, users can specify the network zone they wish to redirect, allowing greater control over traffic management.

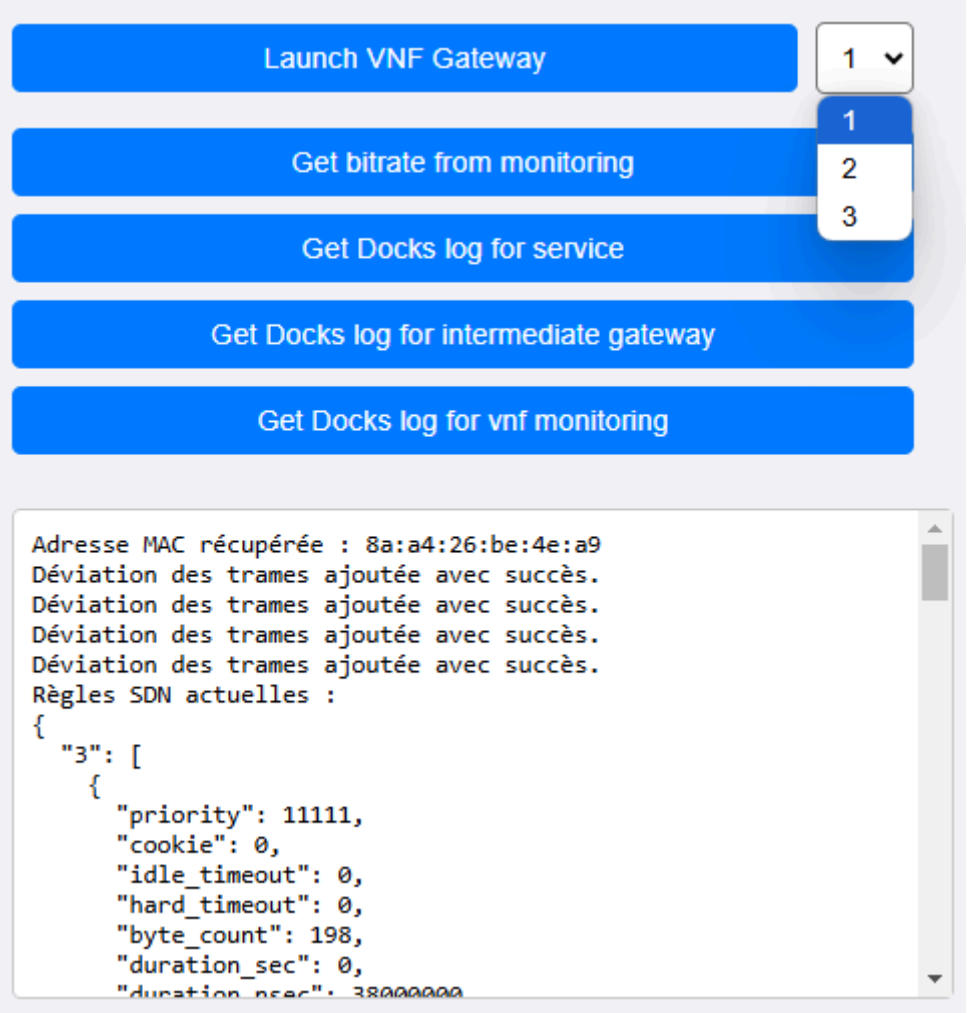

Figure 4. User interface with “Launch VNF Gateway” button

Interface Limitations:

- **Sequential VNF Launch**: The UI requires the user to start the VNFs in a specific order. This prevents errors, as starting VNFs in the wrong sequence could disrupt the SDN rules.

In summary, while the interface enhances usability, it still imposes a structured process for managing VNFs. This approach ensures smooth operation by guiding the user through required steps, particularly with traffic redirection and log retrieval.

Potential Enhancements

Given more time, several enhancements could have been implemented to improve the system's functionality and user experience:

1. **Graphical User Interface**
    - Features of this interface could include:
        - **Real-Time Network Topology Visualization**: Display the current network setup dynamically, showing nodes, links, and traffic flows with a map.
2. **Automation Based on Traffic Thresholds**
    - The system could include a mechanism to automatically create a new intermediate gateway when a specified traffic threshold is exceeded.
    - This automation would help balance network loads without requiring manual intervention, ensuring smoother operation.

These features would streamline the process, making the system more user-friendly and adaptable while improving overall performance and scalability.

Conclusion

In conclusion, addressing network saturation effectively requires meticulous traffic monitoring, dynamic redirection capabilities, and scalable system designs. Through the deployment of monitoring and intermediate gateway VNF, this solution demonstrates the potential to optimize network traffic and improve responsiveness. While this approach has proven effective, opportunities for future enhancements, such as real-time visualizations and automated responses to traffic surges, can further refine the system’s utility. Ultimately, this report illustrates how advanced SDN techniques coupled with VNF implementations can significantly improve the efficiency of modern networks while preventing saturation in critical zones.