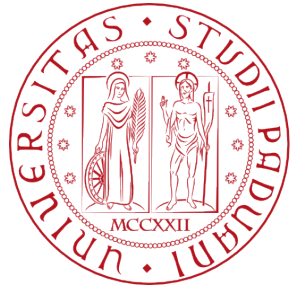
A Lightweight, Container-based approach to Service function chaining

Marco Zanella
1155185
December 2018



Università degli Studi di Padova

Outline

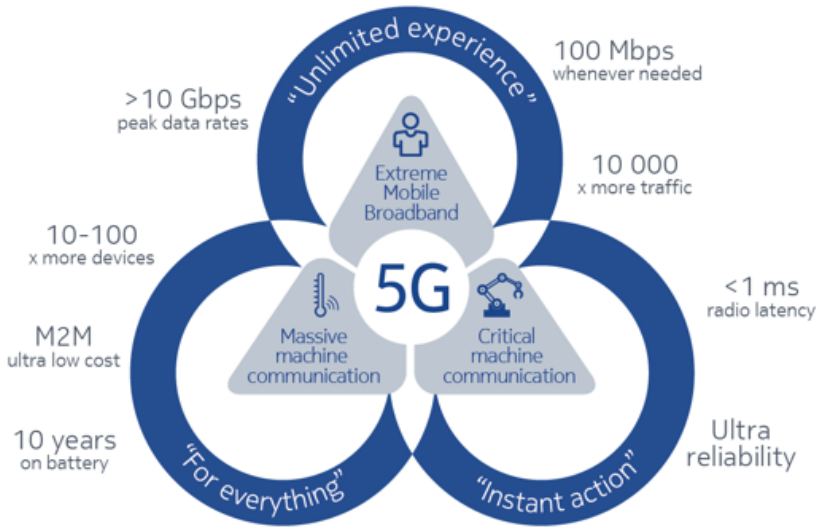
Towards to 5G

The fifth generation of mobile technologies will be relased in 2020 and many changes will came out

It will provide lower latency, better bandwith usage and changes in infrastructures

It is possible thanks to

- Virtualization
- Network slicing
- Software defined networking



Current network infrastructure

To improve performance network function on links are conteplated

This functions are currently provided as a combination of harware and software

This approach cause some drawbacks

- long product cycles
- higher costs
- more difficult updates

Network Function Virtualization

Exploiting virtualization technology is possible to provide function only as software entities called Virtual Network Function (**VNF**)

It allows to remove specific hardware and make it possible to run VNFs on **general purpose** machines

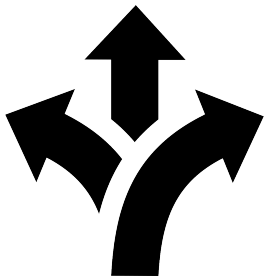
Efficient algorithms and resource management is required to provide same performances as hardware ones

Service Function Chaining

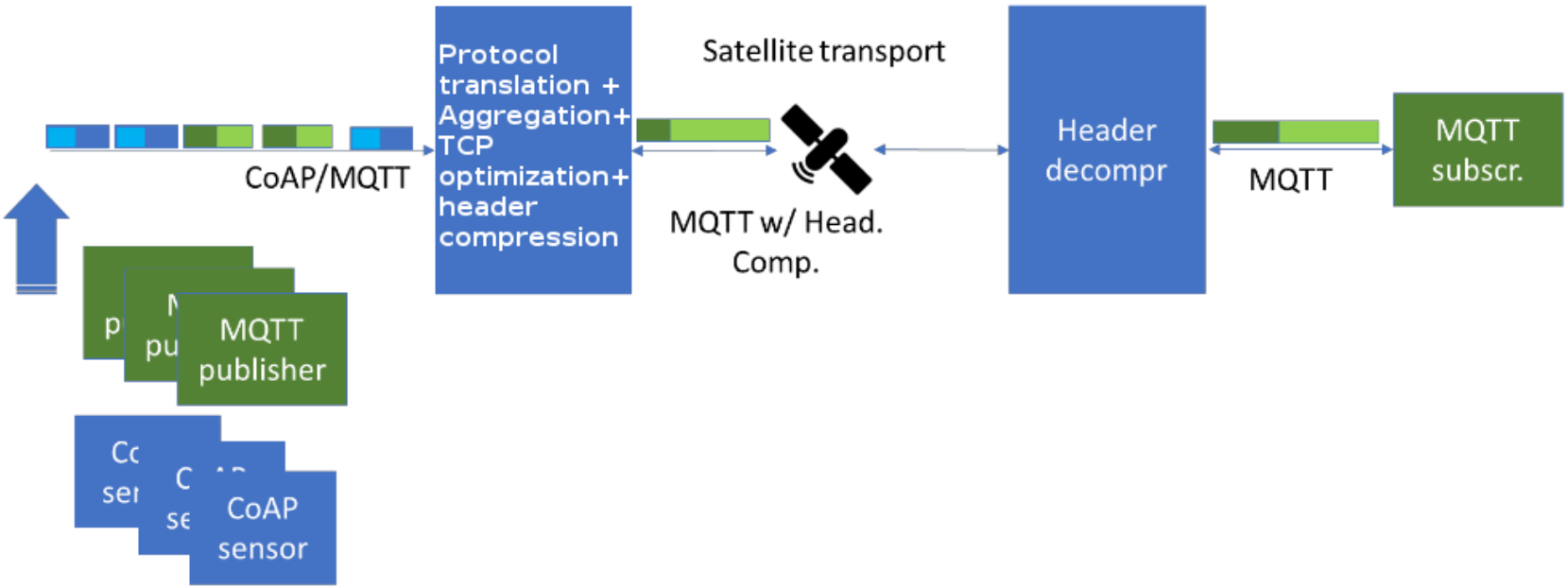
VNF can be combined in sequence to form a chain of transformations

SFCs allow to choose alternative treatments for different traffic types

Traffic can be classified at chain edge and reclassified during chain traverse



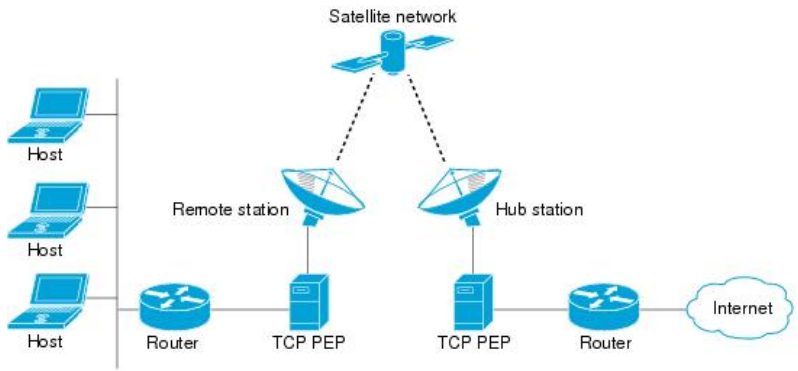
Use case example



This thesis is inspired by VIBeS (*VI*rtualized NFs for *B*roadband *S*atellite networks) project of ESA

The goal is to exploit NFV to enhance communication paths that involve a **satellite** link

Redesign Performance Enhancement Proxies (**PEP**) as virtualized functions



Current VNF/SFC implementations

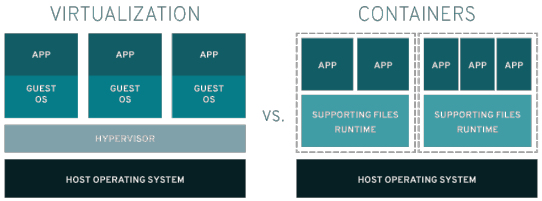
The reference architecture of VNFs, SFCs and their management is ETSI NFV/MANO proposal

Most of the current VNF implementations use **hypervisor-based virtualization**

SFC deployed are **statically** defined at creation time

Our proposal

Exploit operating system virtualization to deploy VNFs



Using Kubernetes as container orchestrator

kubernetes

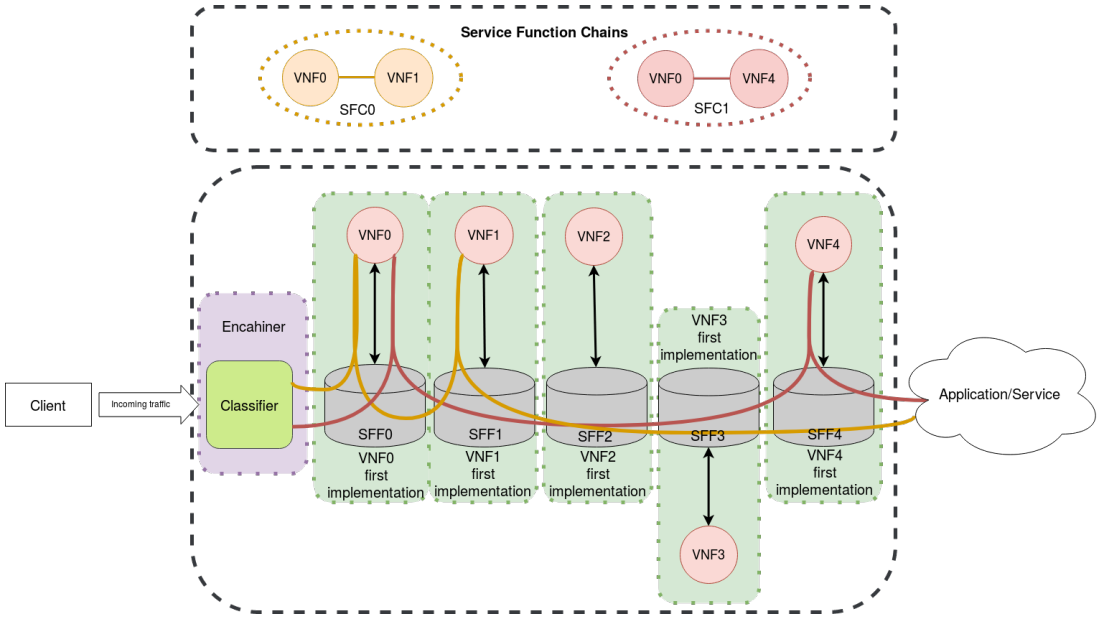
Develop our manager and orchestrator (MANO) for the VNFs/SFCs and the other components to make possible the creation of chains

Evaluate both technologies and approach chosen

Kubernetes

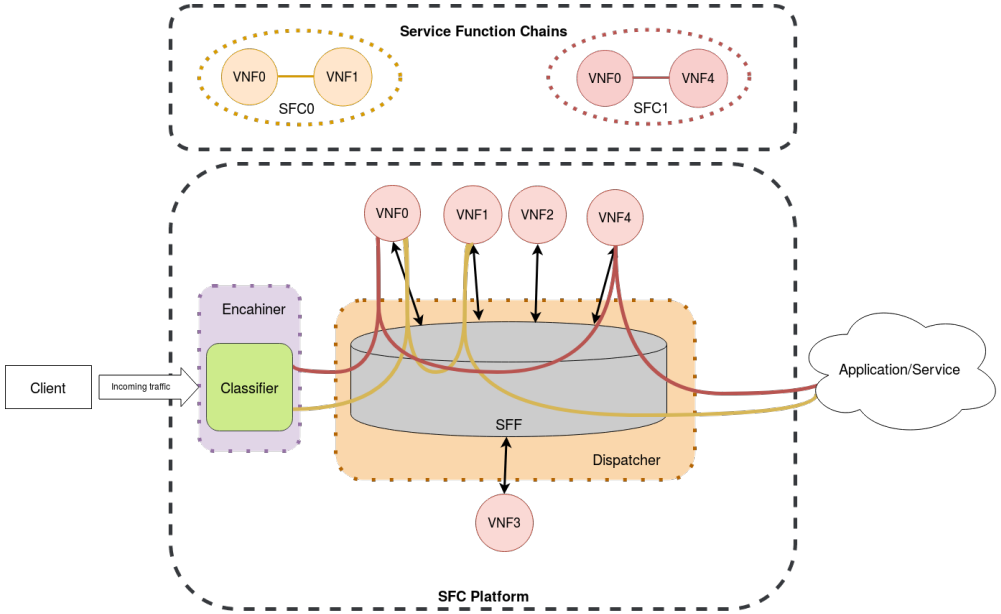
explain what is kubernetes and how we used it

First iteration



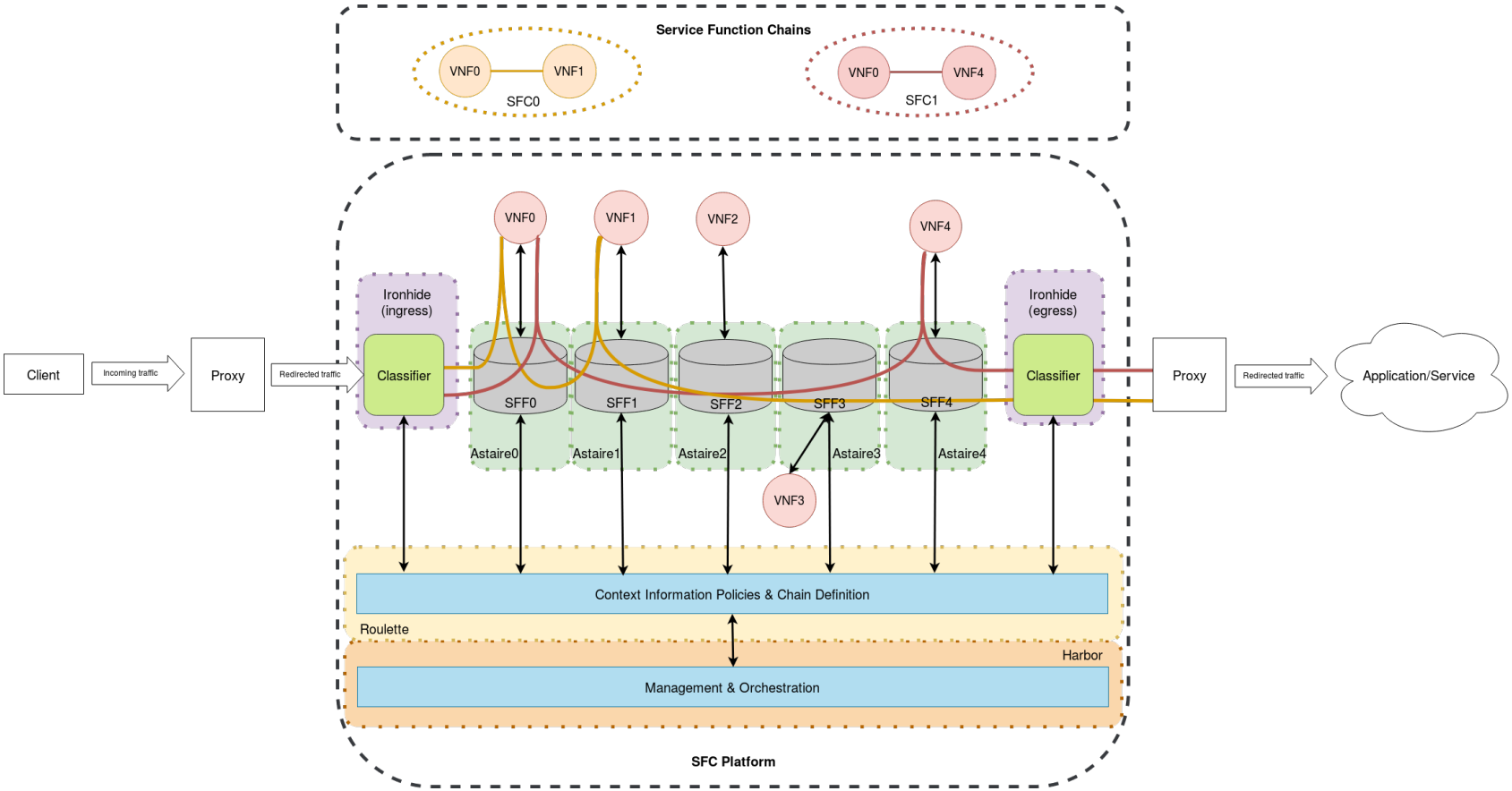
- Entrypoint for the chain to perform classification
- VNF coupled with Service Function Forwarder (SFF) functionality
- SFC chain is statically defined during deployment

Second iteration



- Entrypoint for the chain to perform classification
- VNF decoupled from SFF logic
- Centralized element that talk with each VNF of the chain

Final proposal - High level view



Harbor: Management and Orchestration

Harbor is the **MANO** we developed

It allows to create **definition** of VNFs and SFCs and manages **repositories** on which they are stored

From definition is possible to **run/stop** chains

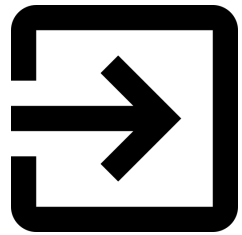
It is not deployed on Kubernetes but it communicate with it

Ironhide: chains endpoints

Ironhide is the component that manages traffic that **enter and exit** from the chain

It classify entering packets chosing the most suitable chain

- Classification based on **transport layer protocol** used
- Performed only at the edges of the chain, not during the traversal



Encapsulate/decapsulate packets to traverse the chain

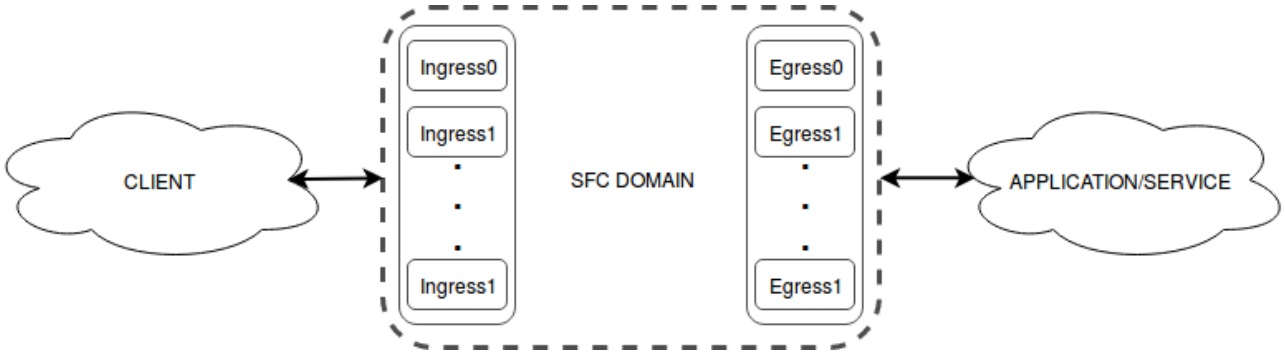
Keep the connection with the sender/receiver of packets

Roulette: Context and Chains provider

Middleware between the SFC components and the MANO

Allows to **access chain definitions** throught APIs

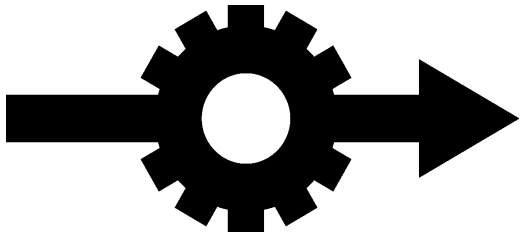
Preserve **end-to-end information** on chains' endpoints used



Astaire: Service Function Forwarder

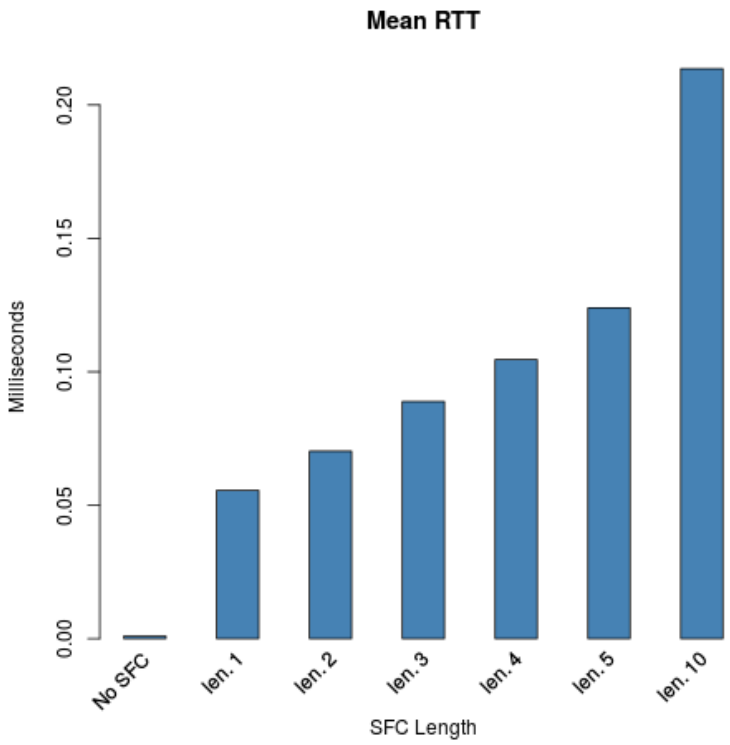
It is the component that logically **create the path** for traffic

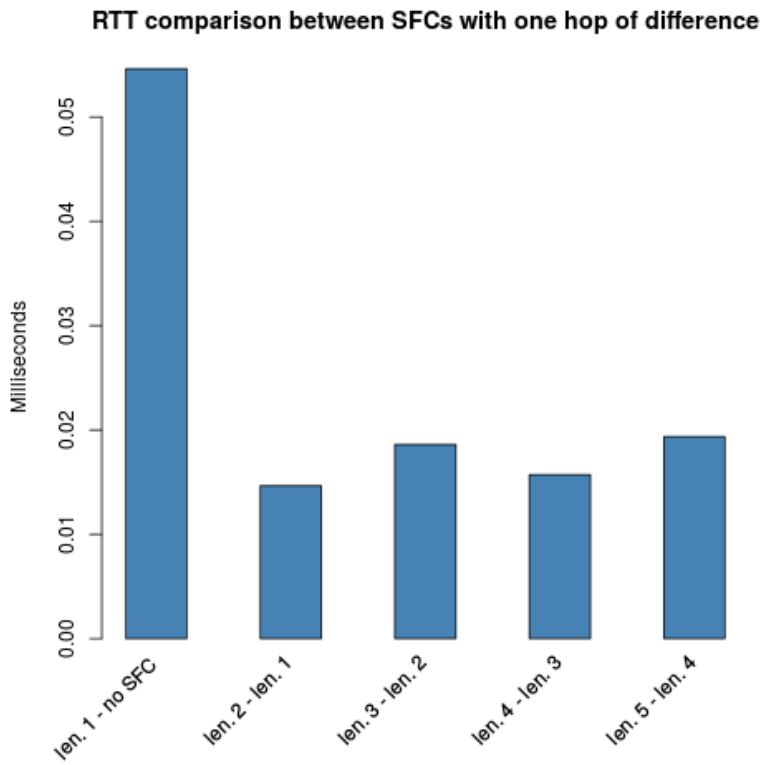
Is accountable to **forward traffic to the VNFs** and manage the response



It queries Roulette to check the next hop based on chain identifier specified in the packet encapsulation

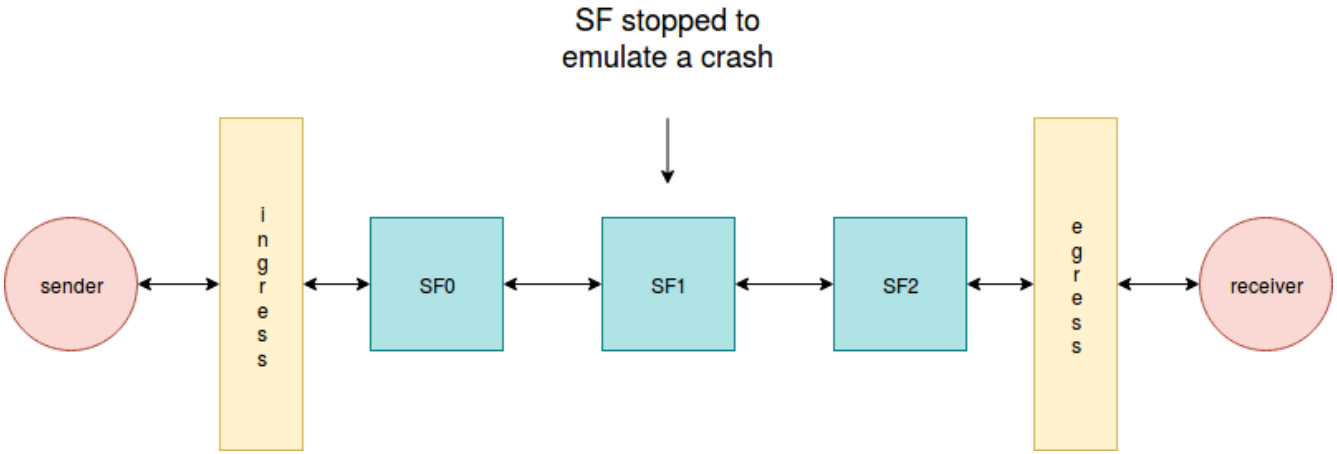
Tests - RTT





- RTT is higher using the SFC platform
- Time is required to elaborate and exchange packets
- Linear increase even due to the usage of the safe function

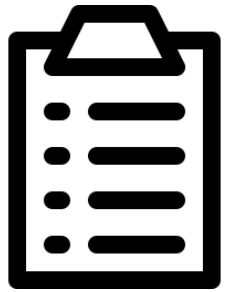
Tests - Time to recover from a fault



- after 5s that the sender starts to send packet the VNF is stopped
- about 13s required to restore a faulty container
- both using TCP and UDP

Future work

Integrate Harbor with Openbaton and expand its features



Rudimental classifier must be refined adding more capabilities

Ironhide implementation requires lower level connectivity to be able to read the whole TCP and UDP headers

Use different transport layer protocol to exchange data into the SFC domain

Conclusion

In this work we developed a proof of concept implementation of an SFC platform

Differently from other current implementation network function and chains are deployed using Kubernetes and Docker

Test the system udnder different conditions

Creating appropriate VNF it makes possible to take advantage of different connection and traffic types