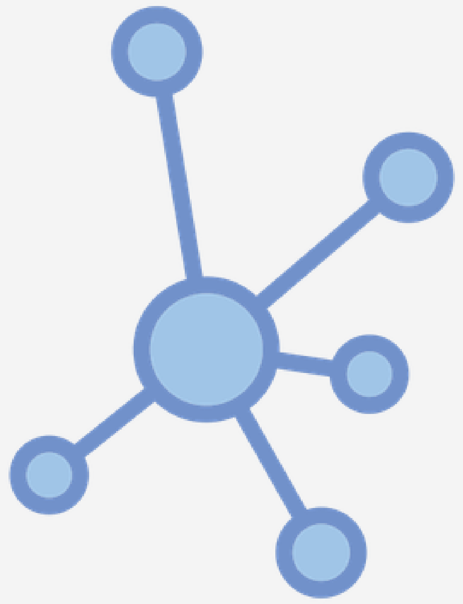
UNIVERSITÀ DI TRENTO

Softwarized and Virtualized mobile networks

Network slice setup optimization



Dennis Cattoni
Marco Lasagna
Andrea Eugenio Cesaretti

Network Slice Setup Optimization

*"**GOAL**: to enable RYU SDN controller to slice the network and then to dynamically re-allocate services in order to maintain desired QoS."*

Comnestsemu

- Comnetsemu repository: https://git.comnets.net/public-repo/comnetsemu

In the initial phase of development, we focused on understanding the ***ComNetsEmu*** environment by studying the codebase of the example scenarios presented during the lectures.

Slices

The subsequent step focused on defining the **network slices**, the associated services, and the QoS policies. In particular, we identified two main slices:

1. Low-Latency Slice
2. High-Throughput Slice

Low-Latency

Goal: The low latency slice is designed to minimize one-way data plane delay between a source host and a destination host within the SDN domain.

Policy:

- 30 ms threshold

Features:

- higher queue priority
- no service migration
- traffic simulated via *iperf3*

High-Throughput

Goal: The high throughput slice is designed to maximize the sustained data rate achievable between a client host and a virtualized service endpoint within the SDN domain.

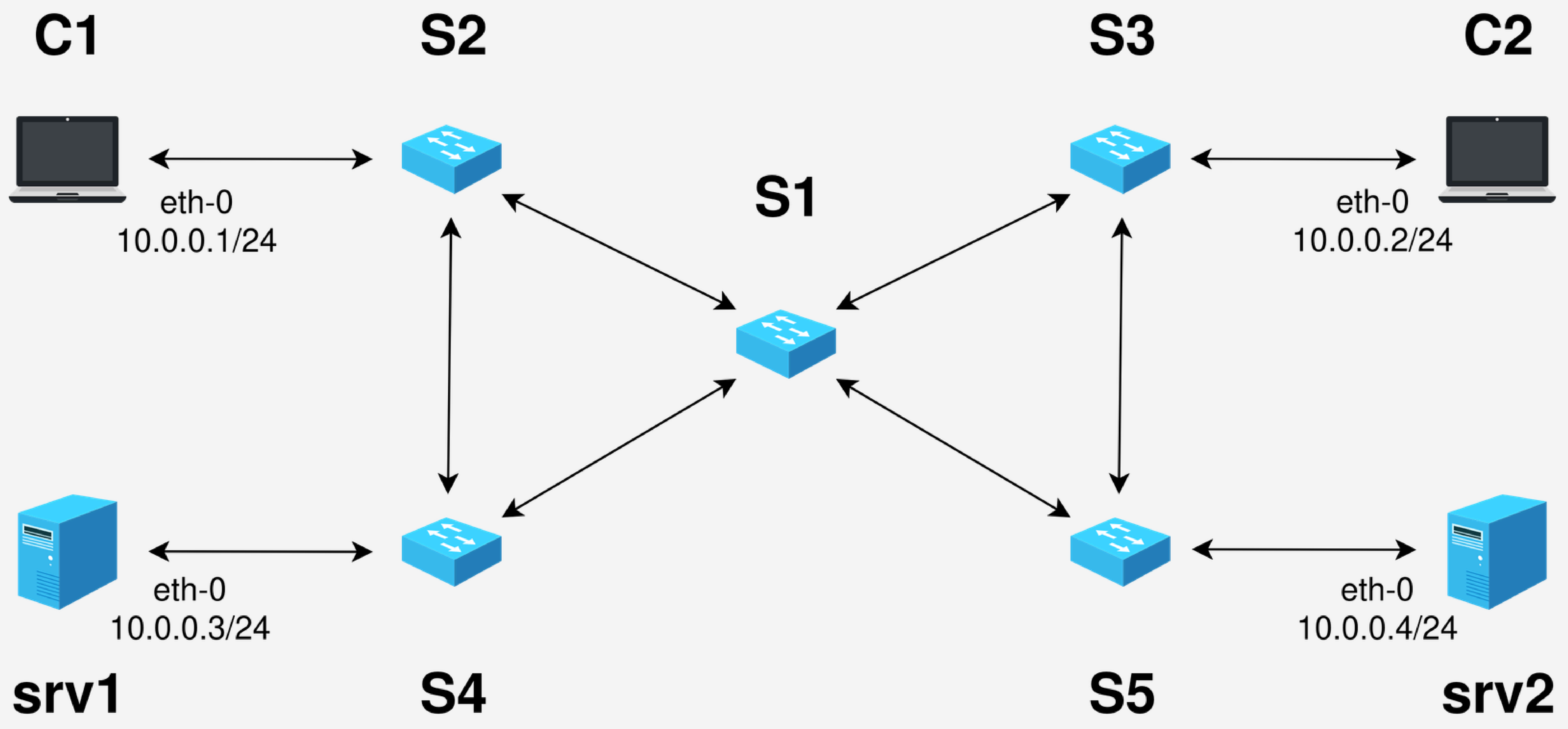
Policy:

- 6 Mbps threshold

Features:

- lower queue priority
- MEC service migration
- custom client and server

Network topology



OpenFlow

- Openflow Doc: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

The next step consisted of analyzing the characteristics of the ***OpenFlow*** southbound interface to identify which events, notifications, state changes, and operational information could be leveraged to design the overall optimization algorithm.

Controller

- RYU controller: https://ryu-sdn.org/

Within the SDN architecture, **Ryu** operates as the SDN controller, implementing the control plane logic. The next step was to design and implement a custom Ryu controller from scratch in order to address the QoS optimization requirements of the defined network slices.

Monitoring

- **QoS Monitoring with Dual-Tier Round-Robin**
  (More details in the full spec)

Goal: maintain an up-to-date network view for slice-aware QoS optimization, while balancing monitoring responsiveness and control-plane overhead.

Monitoring

Monitoring Model: it relies on a periodic thread, which runs every ΔT and collects per-port statistics via OpenFlow *PortStatsRequest* messages.

- **QoS Monitoring with Dual-Tier Round-Robin**
  (More details in the full spec)

Monitoring

Dual-Tier Strategy:

- Tier A – Active Slice Paths
  Poll all switches on active slice paths at every tick
- Tier B – Background Monitoring
  Poll K non-active switches per tick using RR

- **QoS Monitoring with Dual-Tier Round-Robin**
  (More details in the full spec)

Optimization

Low-Latency

It aims to continuously maintain a path that satisfies latency constraints by dynamically monitoring network conditions and triggering event-driven path recomputation upon link failures or QoS violations.

Traffic:

- simulated via *iperf3*

Core feature:

- Higher queue priority

Low-Latency

1. Formally, the optimal path P∗ is defined as:

$$P^* = \arg\min_{P:s\rightsquigarrow d} W_t(P)$$

2. The path cost is the sum of the costs associated with each link at time *t*:

$$W_t(P) = \sum_{i=0}^{n-1} w_t(v_i, v_{i+1})$$

3. The edge cost function is defined as:

$$w_t(u,v) = d(u,v) + \alpha \frac{\rho_t(u,v)}{1 - \rho_t(u,v) + \varepsilon} + p$$

Low-Latency

4. The link utilization ratio between the estimated traffic rate on the link and its nominal capacity:

$$\rho_t(u,v) = \frac{\widehat{x}_t(u,v)}{C(u,v)}$$

5. To reduce the impact of short-term fluctuations and measurement noise, the traffic rate estimate is obtained using an ***EWMA***:

$$\widehat{x}_t(u,v) = \beta \cdot x_t(u,v) + (1-\beta) \cdot \widehat{x}_{t-1}(u,v)$$

where: $x_t(u,v) = \frac{\Delta B_t(u,v) \cdot 8}{\Delta t \cdot 10^6}$

High-Throughput

It aims to maximize sustained data rate by dynamically monitoring path capacity, triggering event-driven path recomputation, and performing service migration when routing alone **cannot** satisfy throughput constraints.

Traffic:

- Streaming on-demand

Core features:

- Widest path
- MEC service migration

High-Throughput

1. Formally, the optimal path P∗ is defined as:

$$P^* = \arg\max_{P:s \rightsquigarrow d} B_t(P)$$

2. The bottleneck throughput is defined as the minimum residual capacity among all links composing the path:

$$B_t(P) = \min_{i=0,\ldots,n-1} r_t(v_i, v_{i+1})$$

3. The residual capacity is computed as:

$$r_t(u,v) = C(u,v) - \widehat{x}_t(u,v)$$

High-Throughput

4. To reduce the impact of short-term fluctuations and measurement noise, the traffic rate estimate is obtained using an ***EWMA***:

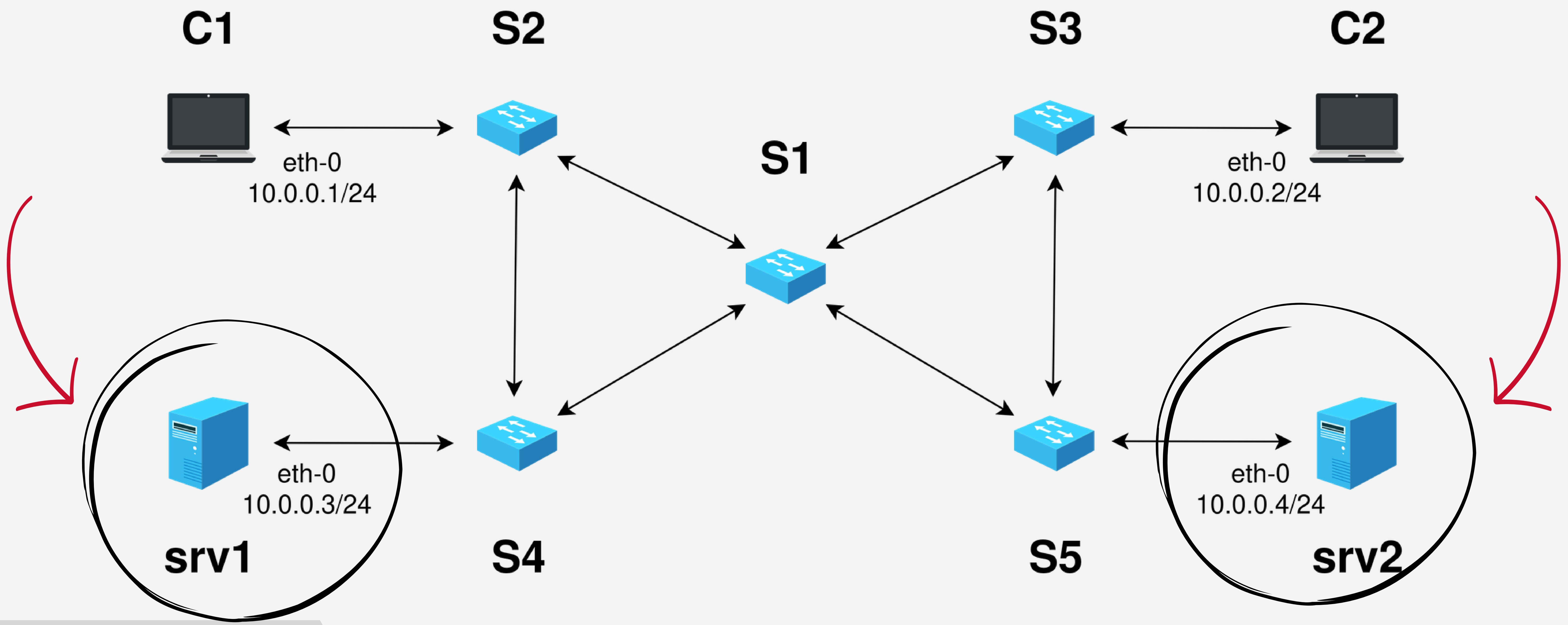
$$\widehat{x}_t(u,v) = \beta \cdot x_t(u,v) + (1-\beta) \cdot \widehat{x}_{t-1}(u,v)$$

where: $$x_t(u,v) = \frac{\Delta B_t(u,v) \cdot 8}{\Delta t \cdot 10^6}$$

Service migration

MEC is leveraged as a complementary mechanism to SDN routing: when the throughput constraint cannot be satisfied through path recomputation, the *Orchestrator* instructs the *Controller* to migrate the service to an alternative backend on an edge node closer to the end user.

Network topology



Almost done…

Img source: https://deeprockgalactic.wiki.gg/wiki/APD-B317

NFV

- Docker: https://www.docker.com/

As a next step, **NFV** is used to virtualize the streaming on-demand service as a containerized network function, enabling controller-driven service migration that is invisible to the client, thanks to SDN-based VIP/VMAC and NAT.

Demo time!

Limitations

Known Limitations
(More details in the full doc)

Environment Constraints:
the experimental setup is bound to ComNetsEmu on Ubuntu 20.04 LTS.

Scalability Trade-offs:
balanced monitoring limits overhead but delays visibility on non-active links.

Security:
it was not taken as a primary design goal.

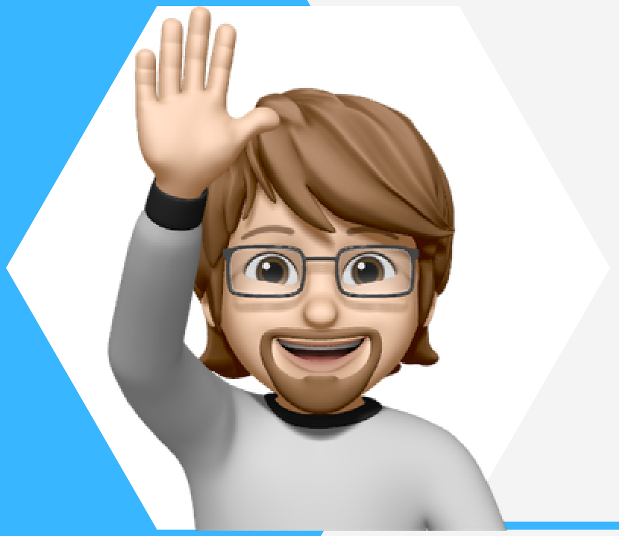
Future work

Future work

(More details in the full doc)

Future work includes large-scale scalability evaluation, integration of modern transport protocols such as QUIC, and the incorporation of security-aware control policies and threat mitigation mechanisms.

Thank you for your attention



Marco Lasagna

Dennis Cattoni

Andrea Eugenio Cesaretti