atlanTTic
research center
for Telecommunication Technologies

The University of Vigo in OAI: Virtual Operators

Felipe Gil-Castiñeira (felipe@uvigo.es)

XUNTA DE GALICIA

UNIÓN EUROPEA

Universidad de Vigo

galicia

The University of Vigo and atlanTTic

atlantTTic is the Research Center for Telecommunication Technologies of University of Vigo

- 11 research groups.
- Specialized in Telecommunication and ICT.

2016

Capacities

165 RESEARCH STAFF

- 16 POSTDOC
- 38 PREDOC
- 32 TECHNICAL STAFF

Research activity and scientific production

- **4,4M€** R&D FUNDS
- **176** PUBLICATIONS TOTAL (WOS O SCOPUS)
- **42** PAPERS Q1

VALORIZATION AND TRANSFER

39 ACTIVE CONTRACTS

2,38M € R&D AND TRANSFER CONTRACTS

International collaborative projects

176 (42 Q1) papers coauthored with foreign researchers 2016

4 H2020

Last 2 years (2015-2016)

Relations with Industry and Technology Transfer

GTI Research group

An experienced team

- 8 professors.
- 3 postdocs.
- 9 telecommunication engineers.
- For several years “First ICT group of the University of Vigo in knowledge transference”.
- Researching in 5G (mainly core), embedded systems, and IoT.
- Teaching in “Wireless Networks”, “Embedded Systems”, “Mobile Apps”, “Cybersecurity”.

How are we using OAI?

Network slicing for virtual operators

SDN-enabled converged network optimization

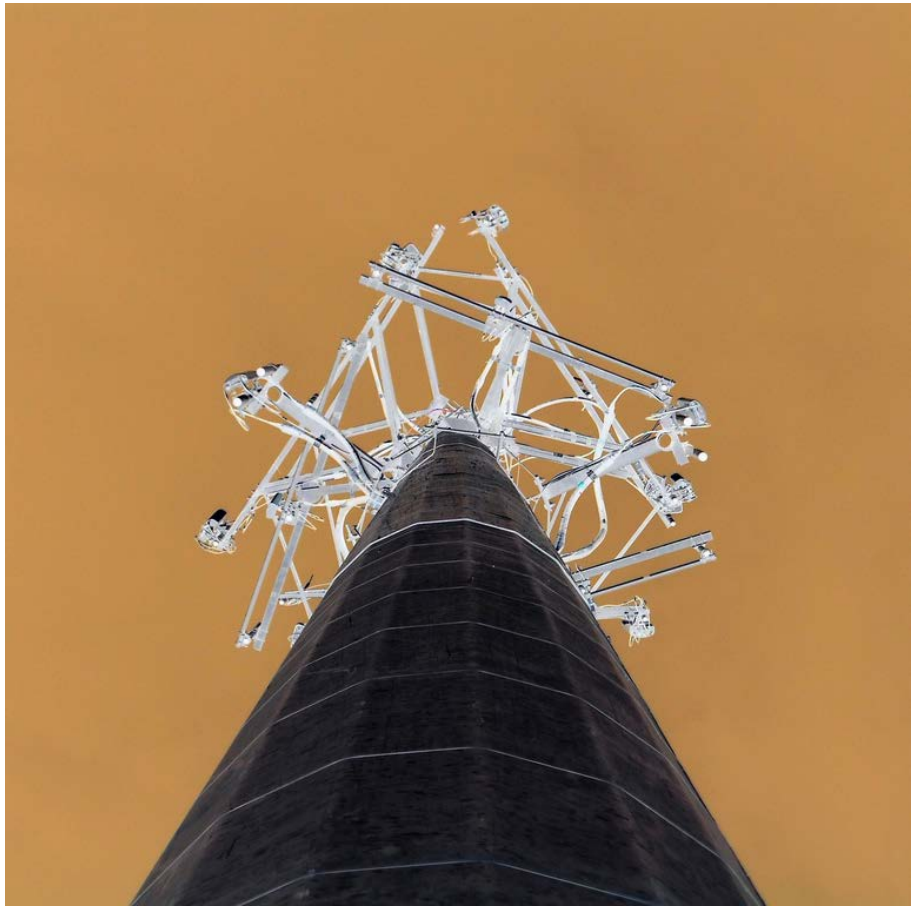
- SDN agents in users' terminals.
- The network handles terminal connections directly.

RAN slicing (sharing):

- Several operators using the same SDR radio interface (e.g. frequency multiplexing).

Flexible core:

- Virtualized EPC using microservices.
- Dynamic core instances.
- Dynamic migration (e.g. moving network functions from core to the edge).
    - GTP Tunnels in OAI.
    - Integrating OAI and OpenEPC.

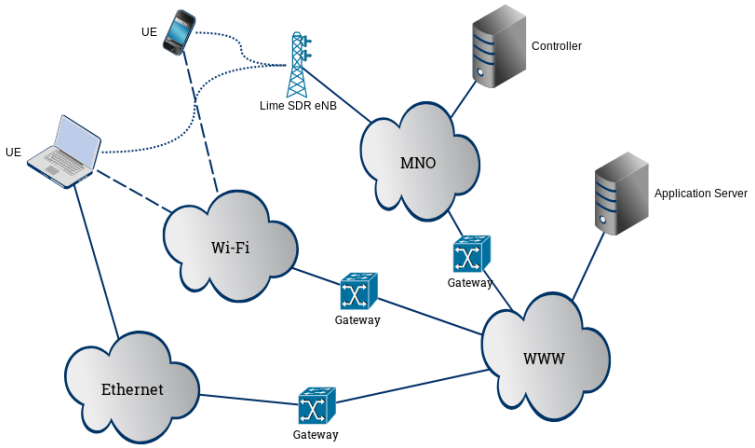


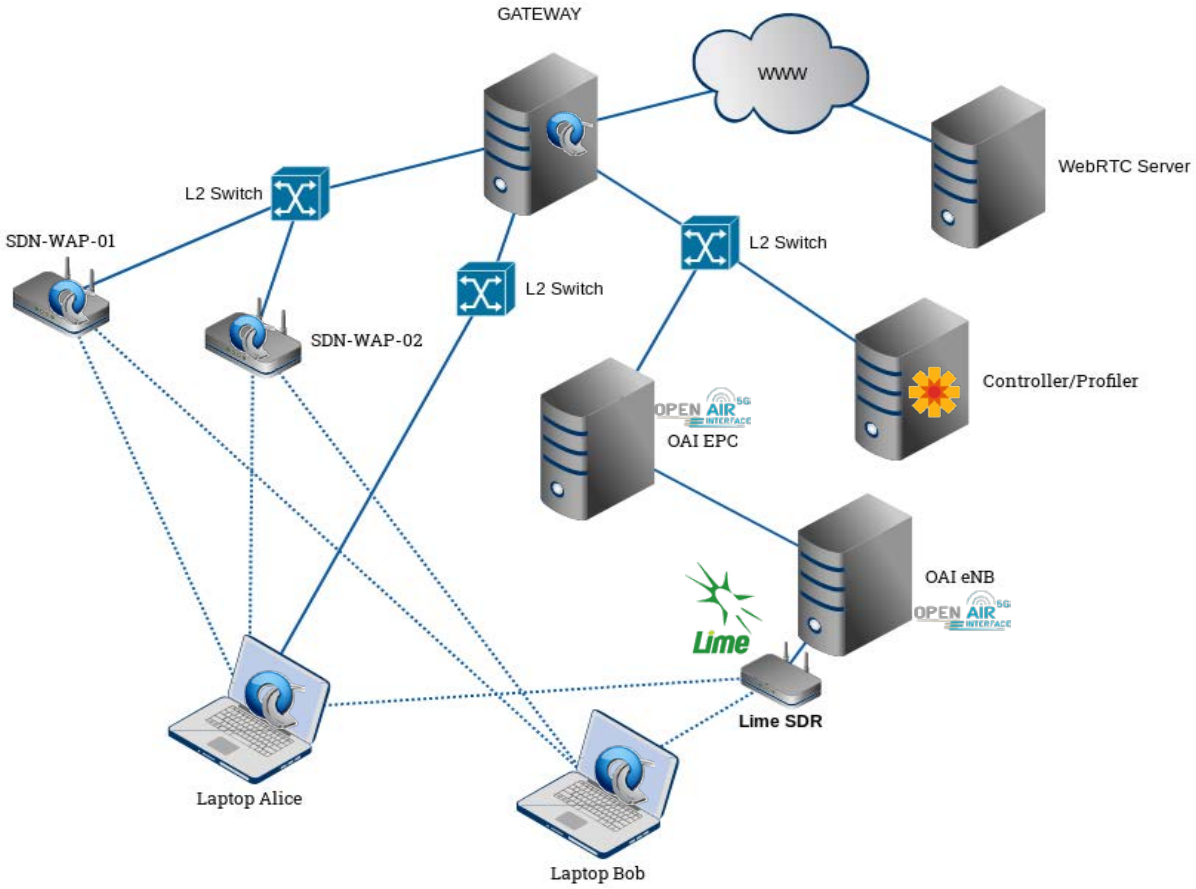
SDN-enabled converged network optimization

SDN Enabled terminals (I)

Terminal connectivity managed from the core

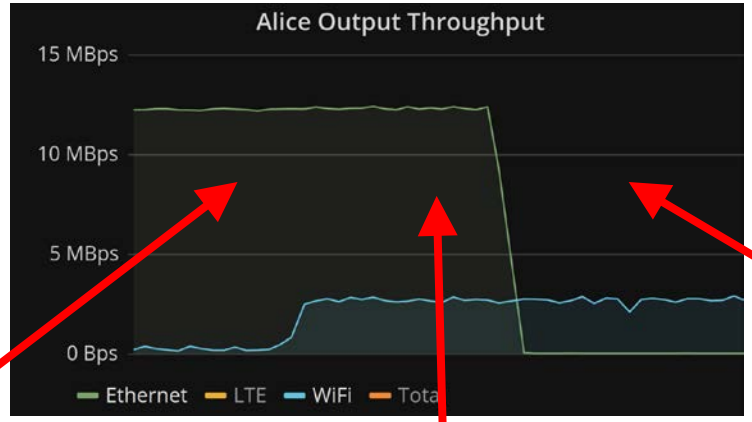
- A terminal is commanded by the controller to switch between two Wireless Access Points without losing session continuity.
- The LTE network (created with OpenAirInterface) is used to control the user terminal.

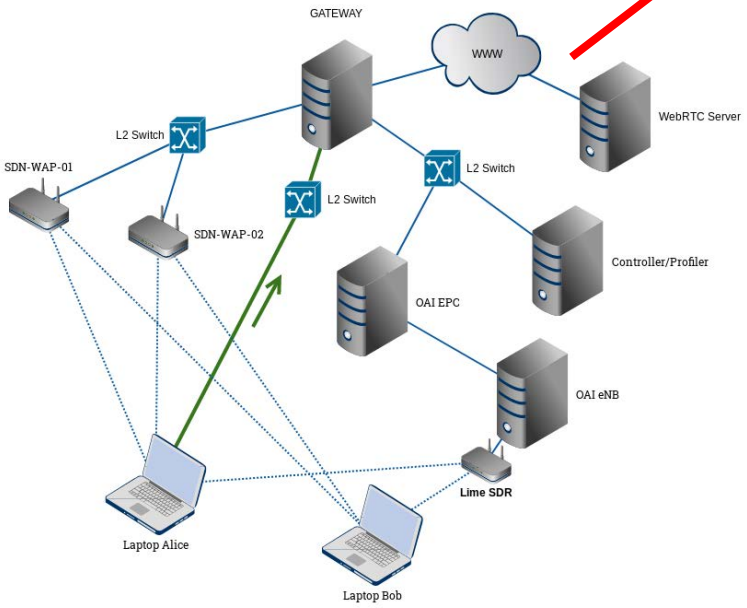


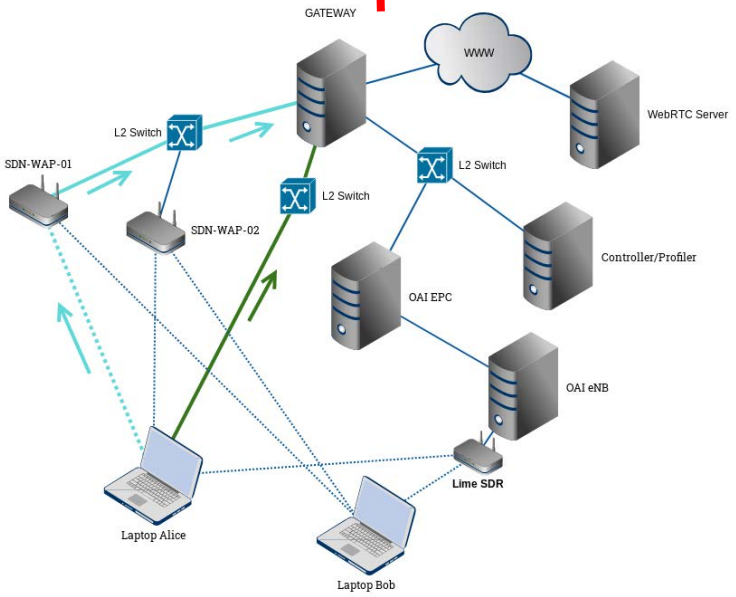


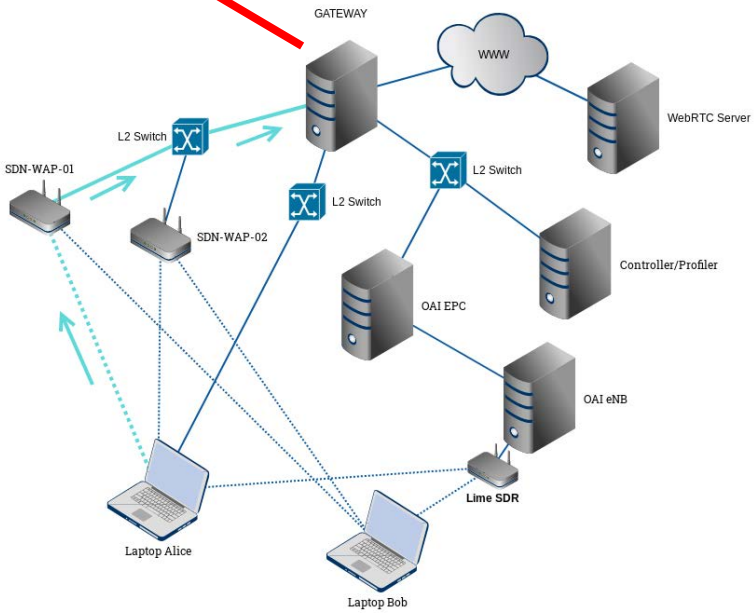
SDN Enabled terminals (II)

Traffic Steering



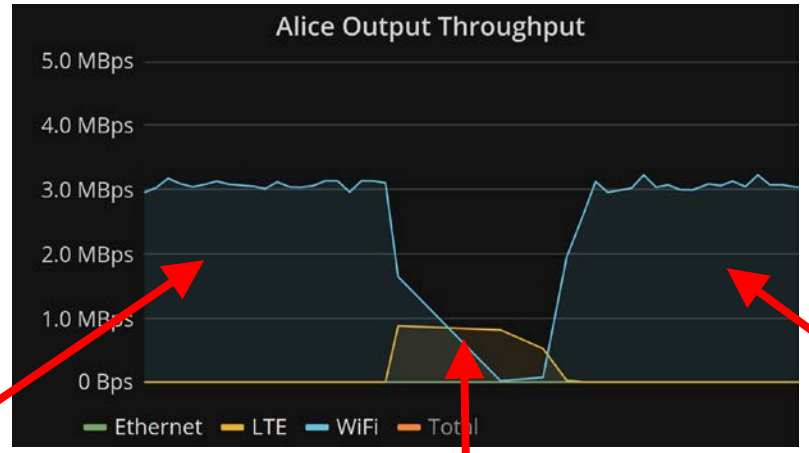


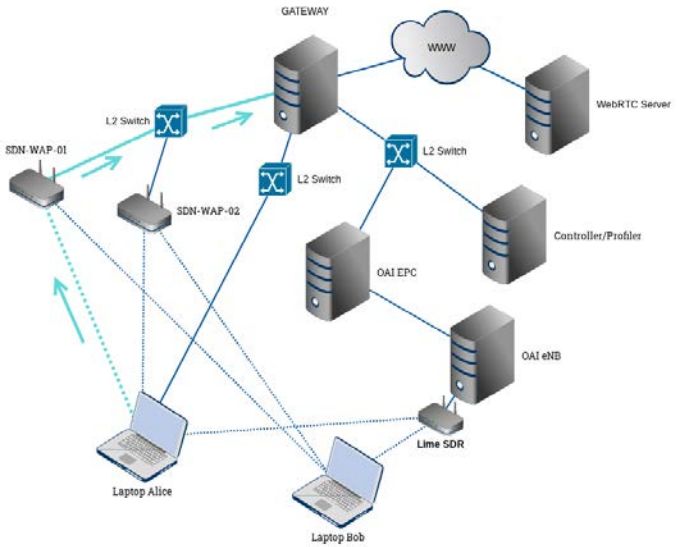


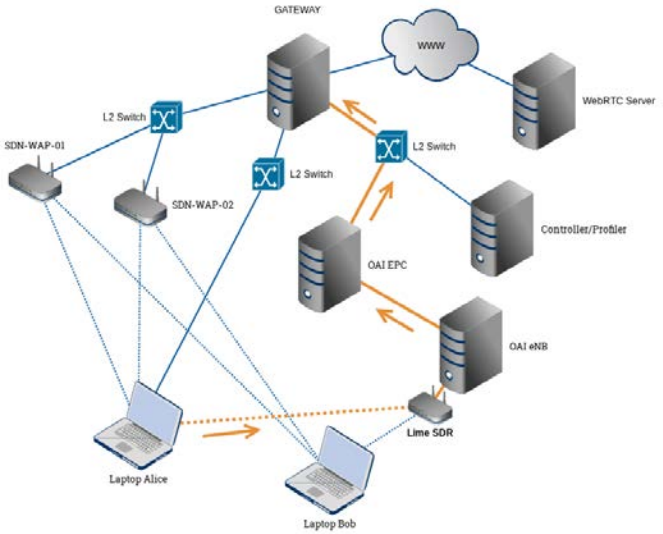


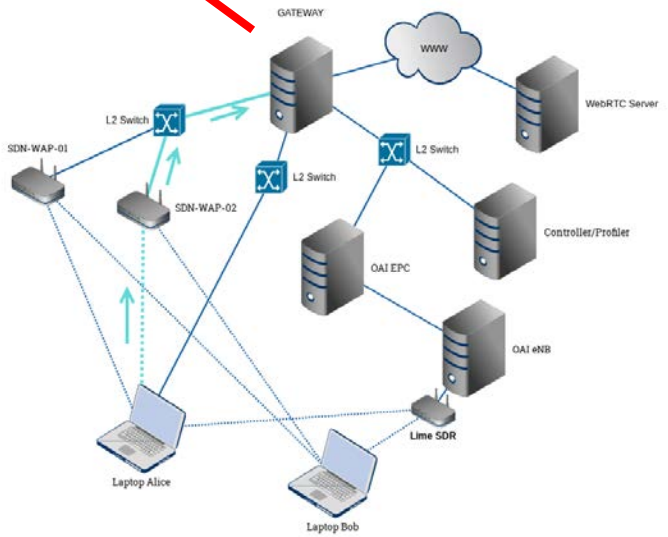
SDN Enabled terminals (III)

Session continuity









SDN Enabled terminals (IV)

Session continuity with WebRTC

- We have checked the operation of the “session continuity” mechanism with WebRTC videoconferences

Tools used

- OpenAirInterface EPC core network.
- OpenAirInterface eNodeB lte-softmodem.
- OpenDayLight for the SDN controller implemented in Apache Karaf OSGi Platform.
- MySQL Server for the Profiler database.
- Oracle Java 8 platform for the implementation of the client daemons.
- WebRTC for the video streaming.
- iPerf for the throughput test.
- Grafana with InfluxDB for the presentation of measurements.
- Wireshark for development.

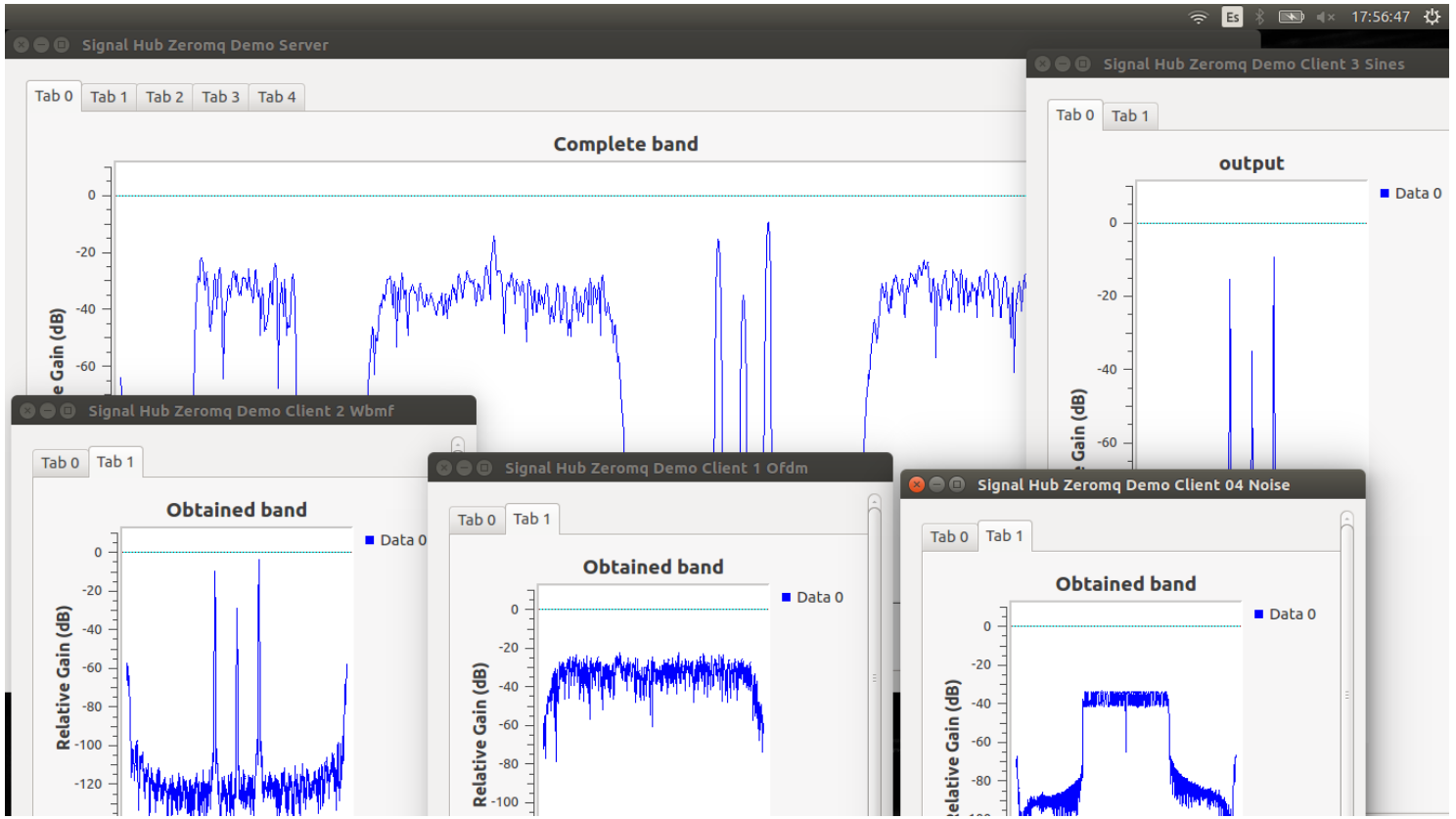
RAN Sharing

RAN Slicing Implementation using OAI (I)

Summary

Two implementations:

- Time sharing.
- Frequency sharing:
  - First test using GNU Radio.
  - Work in progress with OAI.

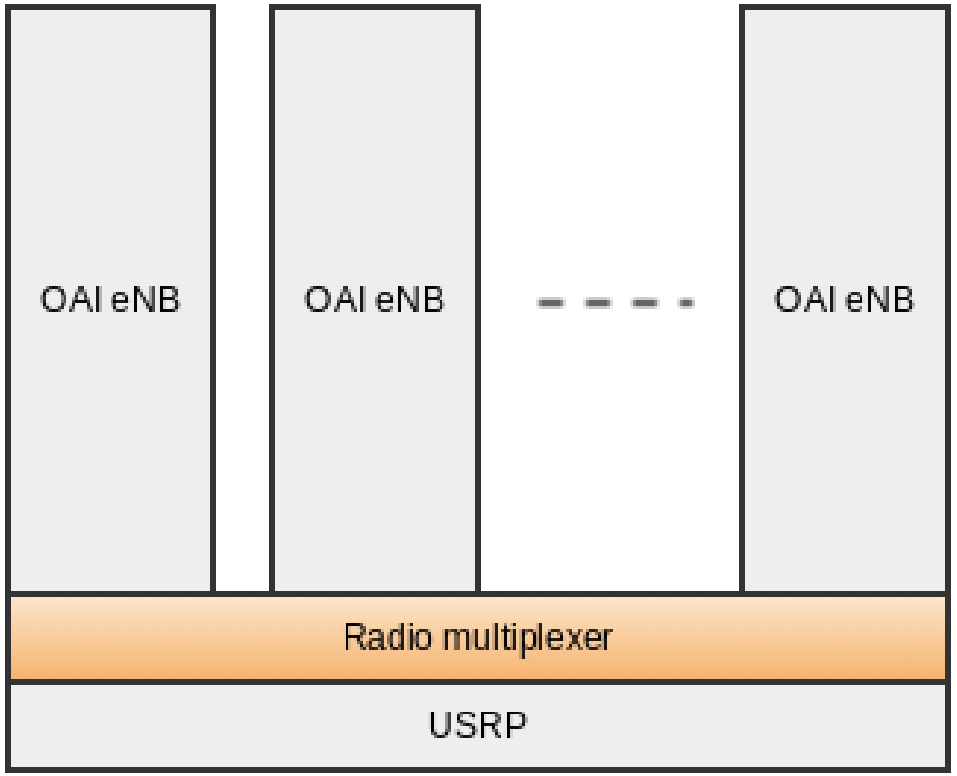


RAN Slicing Implementation using OAI (II)

What do we mean with RAN Sharing?

Enable different eNB instances (or other technologies) **to use the same radio (USRP B210).**

- This means one eNB at a time (time sharing) or using different frequencies for each eNB.
- We need a “multiplexer”.



RAN Slicing Implementation using OAI (III)

Time Sharing

Why do we need Time Sharing?

- With time sharing one can use the whole radio resource when it's needed and release it when not.
- In the context of "virtual operators", it could happen a network/cell is only necessary at certain hours of the day.

We can do this in OAI right now… but it's slow:

- To share the same USRP, we have first to stop an eNB and start the one that we are going to use.
- We have been working on this issue and we have reduced the time for the transition.
  - Dynamic association B210 ←→ eNB.

```
// Initialize USRP device
device->openair0_cfg = openair0_cfg;

std::string args = "type=b200";
uhd::device_addrs_t device_adds = uhd

int vers=0,subvers=0,subsubvers=0;
int bw_gain_adjust=0;

sscanf(uhd::get_version_string().c_st
LOG_I(PHY,"Checking for USRPs : UHD %
      uhd::get_version_string().c_str
```

RAN Slicing Implementation using OAI (IV)

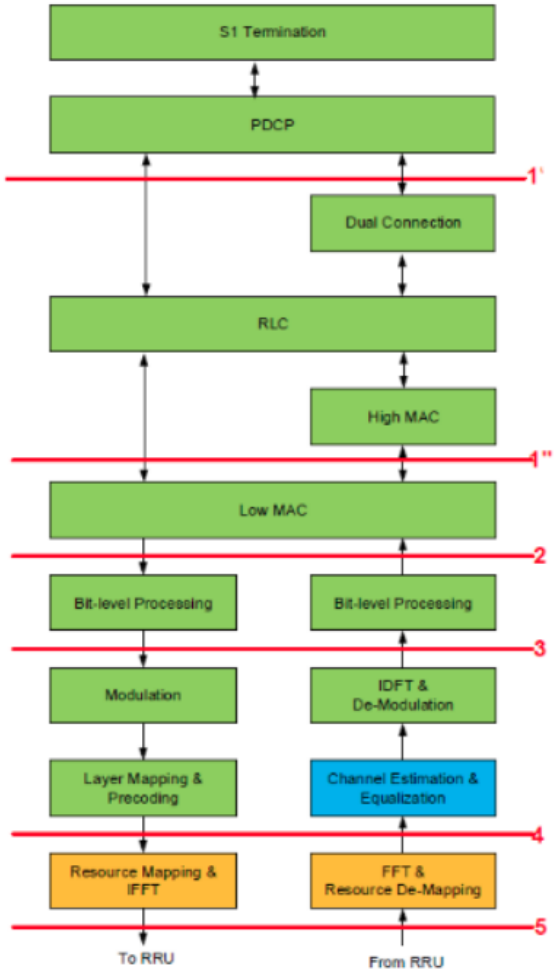
Frequency Sharing

Why do we need "frequency sharing"?

- Finer grained usage of the radio medium resource.
- We can run several eNBs on the same radio device.

Frequency sharing may be performed at multiple levels in the stack (IF5, IF4.5 or even higher).

- We have begun by researching feasibility at the lower level (nearest to the USRP).
- Problem: The USRP only accepts one pair of I/Q flows per channel (for RX and for TX).

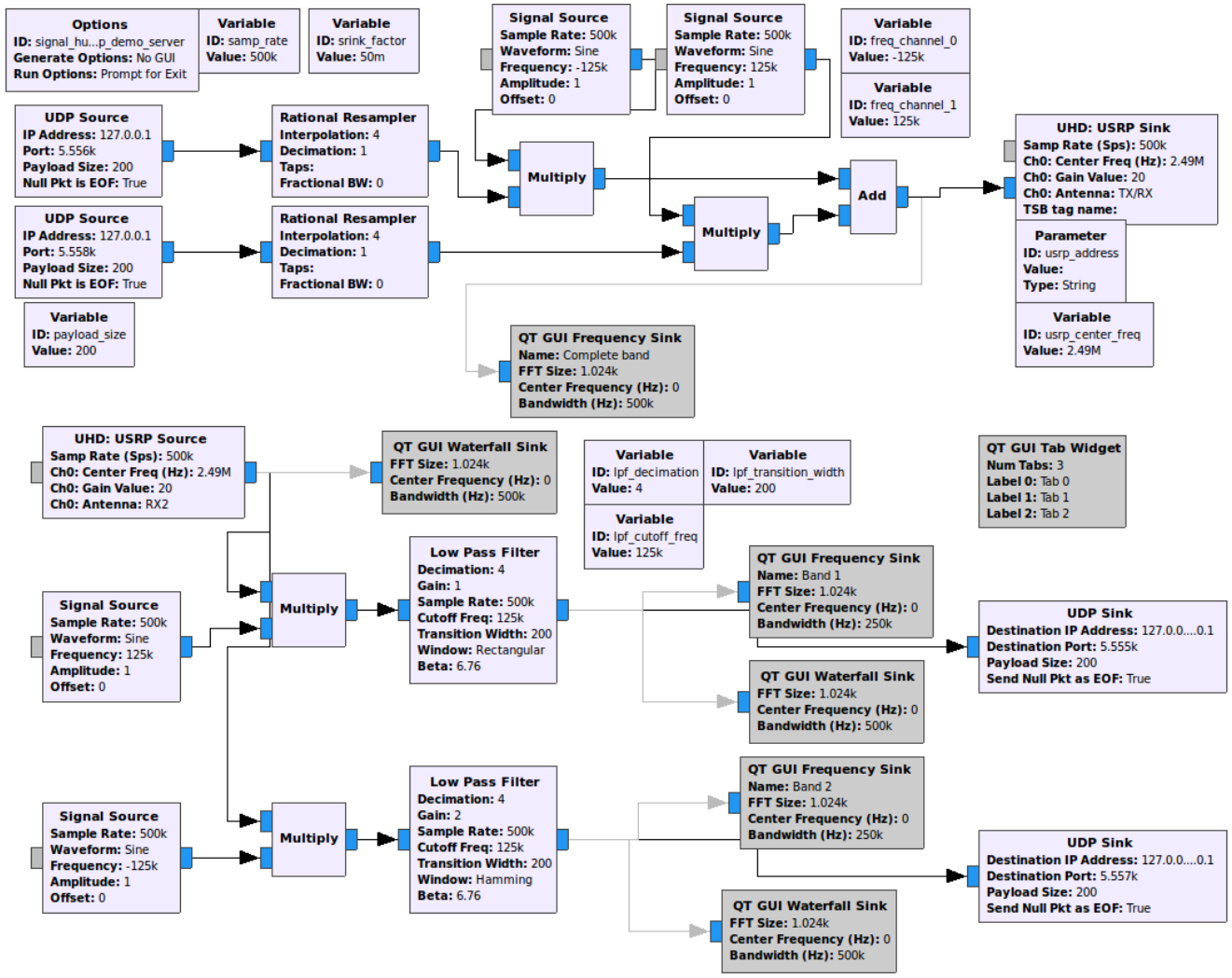


IEEE 1914 NGFI functional splits

RAN Slicing Implementation using OAI (V)

Frequency Sharing Implementation: GNU Radio Tests

- We have some GNU Radio developments that permit RX/TX for two I/Q flows.
- The channels have to be mixed in frequency in TX and filtered in RX.
- We have used UDP and ZeroMQ for I/Q sources/sinks.
- We can share the USRP, but it's CPU consuming.



RAN Slicing Implementation using OAI (VI)

Frequency Sharing Implementation: C++ implementation

Currently, we are working in a C++ implementation of the GNU Radio version:

- We prefer not to be tied to the GPL license of GNU Radio if we want to contribute back to OAI.
- We think that we can achieve a better performance by writing a simple application.

We are trying to improve performance by using the same techniques as OAI:

- Parallelizing with threads.
- Vectorizing code (right now we rely on automatic compiler vectorization).

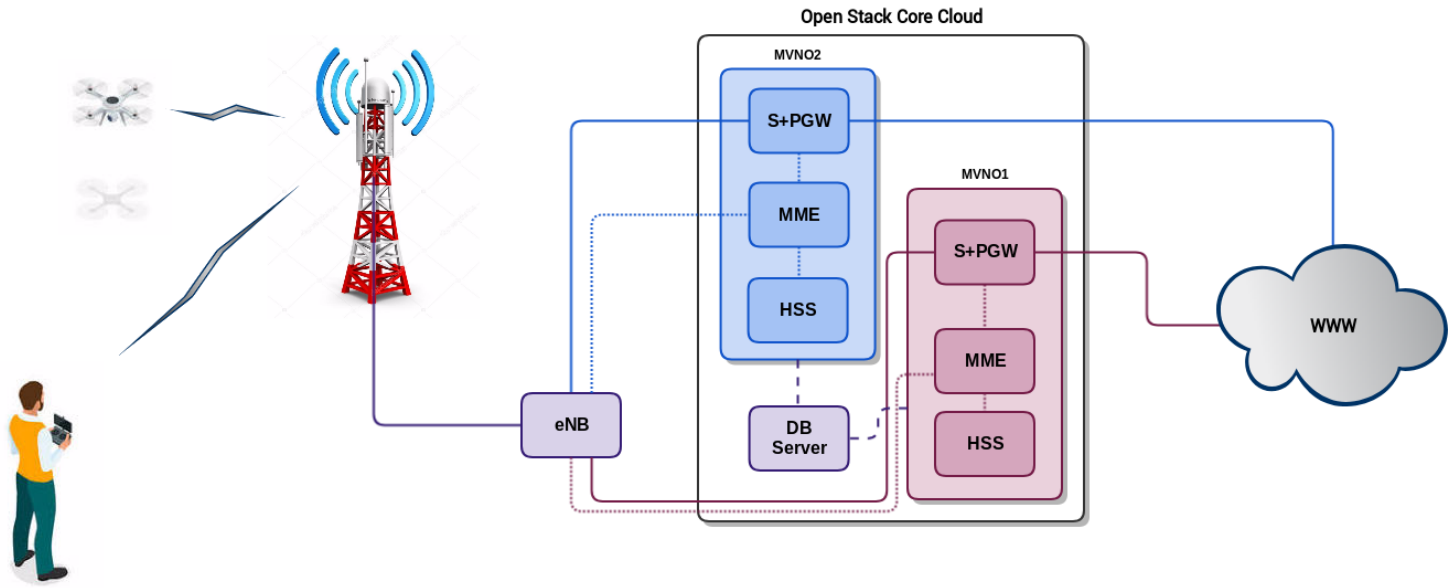
We are experimenting with ZeroMQ sockets to do the job (PUB/SUB architecture).

```
// Allocate Tx buffer
int16_t tx_buff_real[TX_SAMPLES_PER_PACKET] __attribute__ ((aligne
int16_t tx_buff_imag[TX_SAMPLES_PER_PACKET] __attribute__ ((aligne

int loops = TX_LOOPS;
usrp->start_tx();
while (loops > 0){
    loops = loops -1;
    // Generate signal
    signal_1_generator.get_next_samples(TX_SAMPLES_PER_PACKET, a_r
                                        , a_imag);
    signal_2_generator.get_next_samples(TX_SAMPLES_PER_PACKET, b_r
                                        , b_imag);
    snicm_add(a_real, a_imag, b_real, b_imag, signal_real, signal_
                                        , TX_SAMPLES_PER_PACKET);

    // Mix signal to channel 1
    channel_1_generator.get_next_samples(TX_SAMPLES_PER_PACKET, a_
                                        , a_imag);
    snicm_mul(signal_real, signal_imag, a_real, a_imag, tx_buff_re
                                        , tx_buff_imag, TX_SAMPLES_PER_PACKET);

    // Mix signal to channel 2
    channel_2_generator.get_next_samples(TX_SAMPLES_PER_PACKET, a_
                                        , a_imag);
    snicm_mul(signal_real, signal_imag, a_real, a_imag, b_real, b_
                                        , TX_SAMPLES_PER_PACKET);
    snicm_add(b_real, b_imag, tx_buff_real, tx_buff_imag, tx_buff_
                                        , tx_buff_imag, TX_SAMPLES_PER_PACKET);

    // Mix signal to channel 3
    channel_3_generator.get_next_samples(TX_SAMPLES_PER_PACKET, a_
                                        , a_imag);
    snicm_mul(signal_real, signal_imag, a_real, a_imag, b_real, b_
                                        , TX_SAMPLES_PER_PACKET);
```

Flexible core

Flexible core (I)

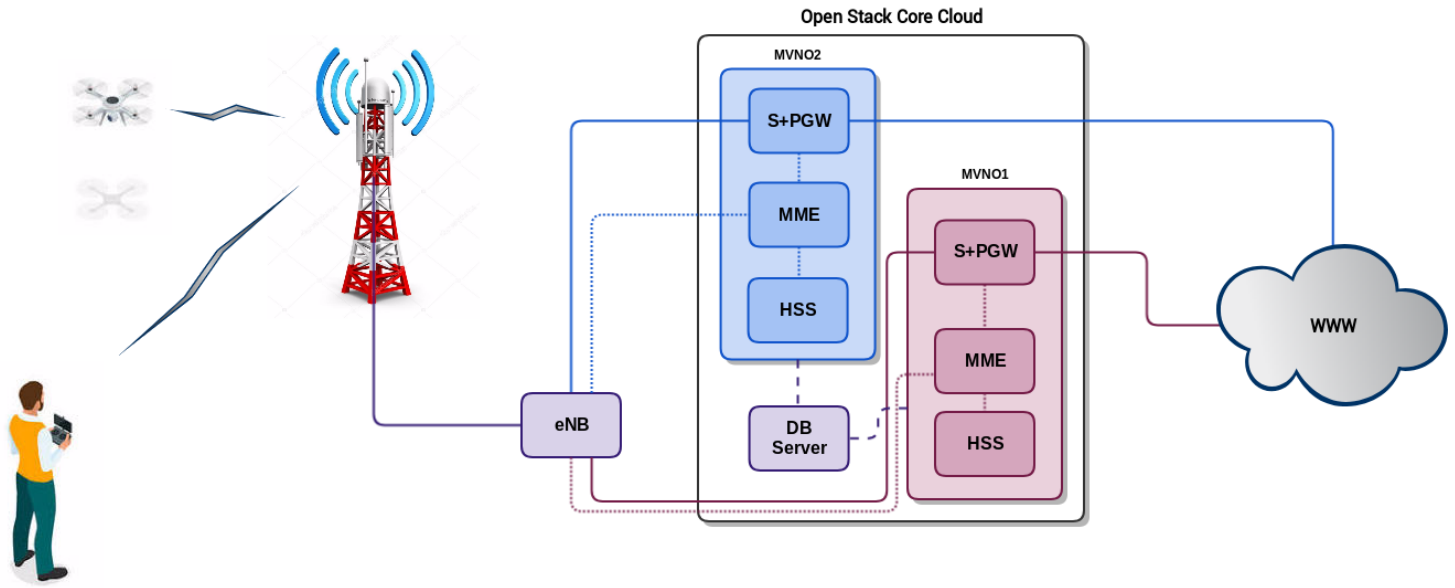
Network slicing for virtual operators



- Specialized virtual operator for the transmission of a video stream (high bandwidth).
- Specialized virtual operator for the transmission of control commands (low latency).

Flexible core (II)

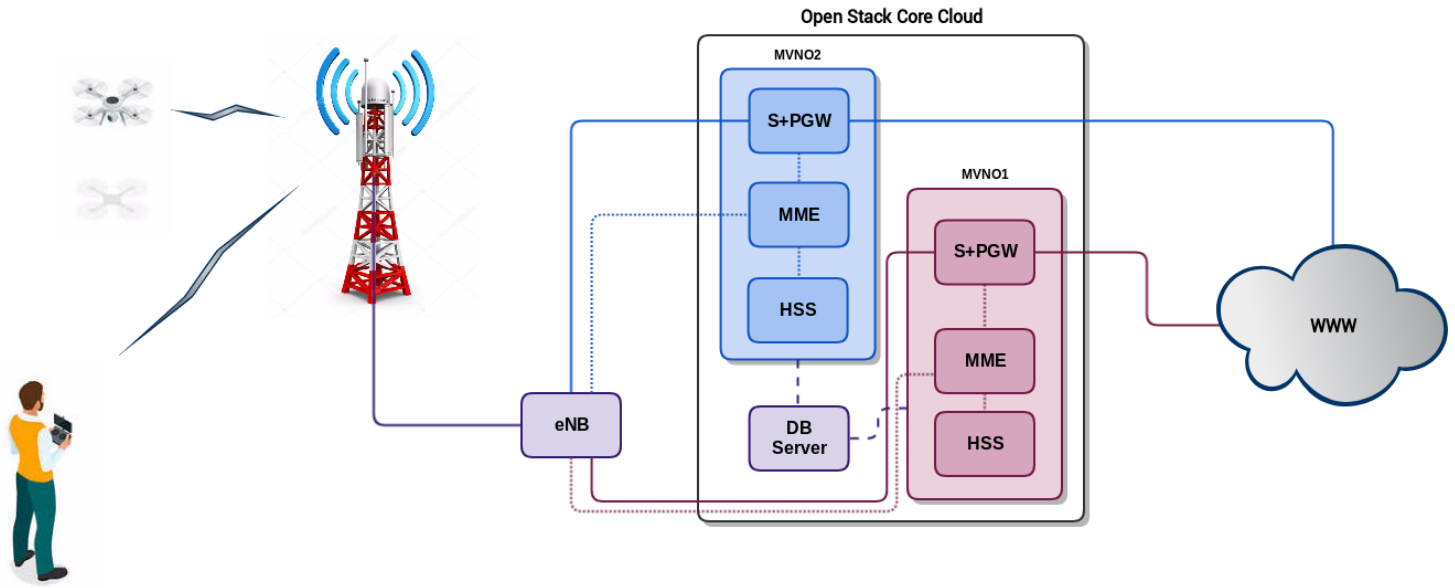
Network slicing for virtual operators



- Parametrized *Heat* template launches HSS, MME & S+PGW.
- EPC modules get their configuration files from a repository through an URL stored in a DB Server.

Flexible core (III)

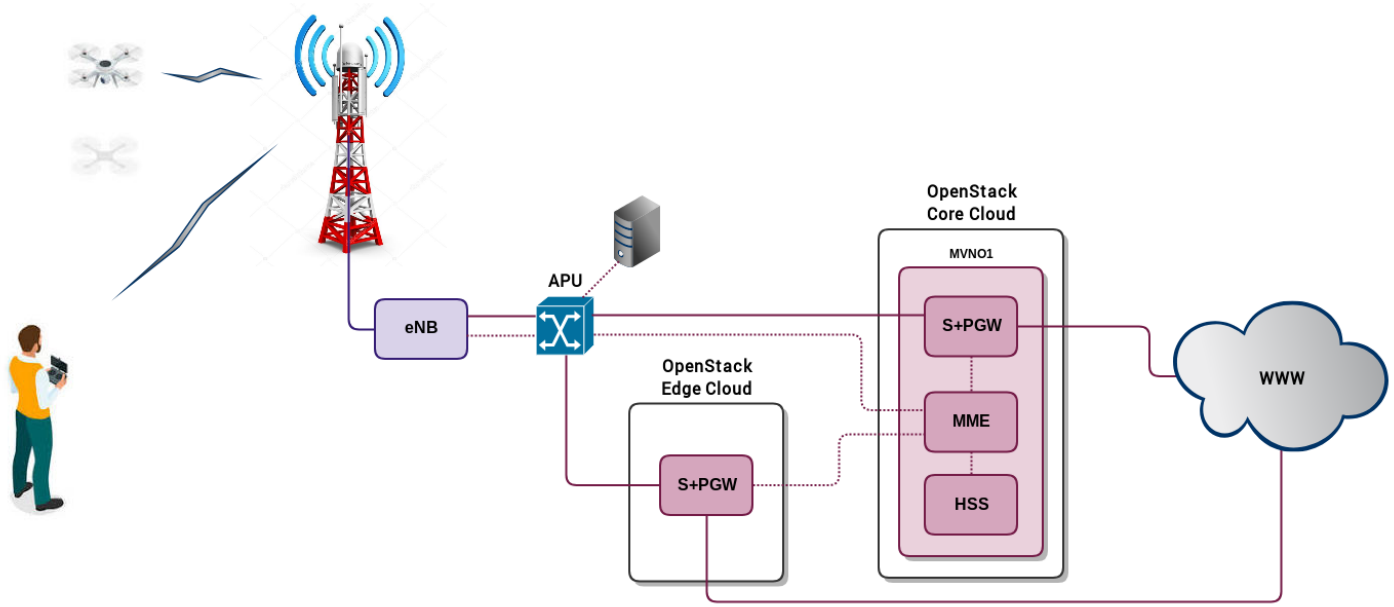
Network slicing for virtual operators



- Tested with two eNBs with our “Time Sharing” implementation.
- The two operators may share the eNodeB (work in progress).

Flexible core (IV)

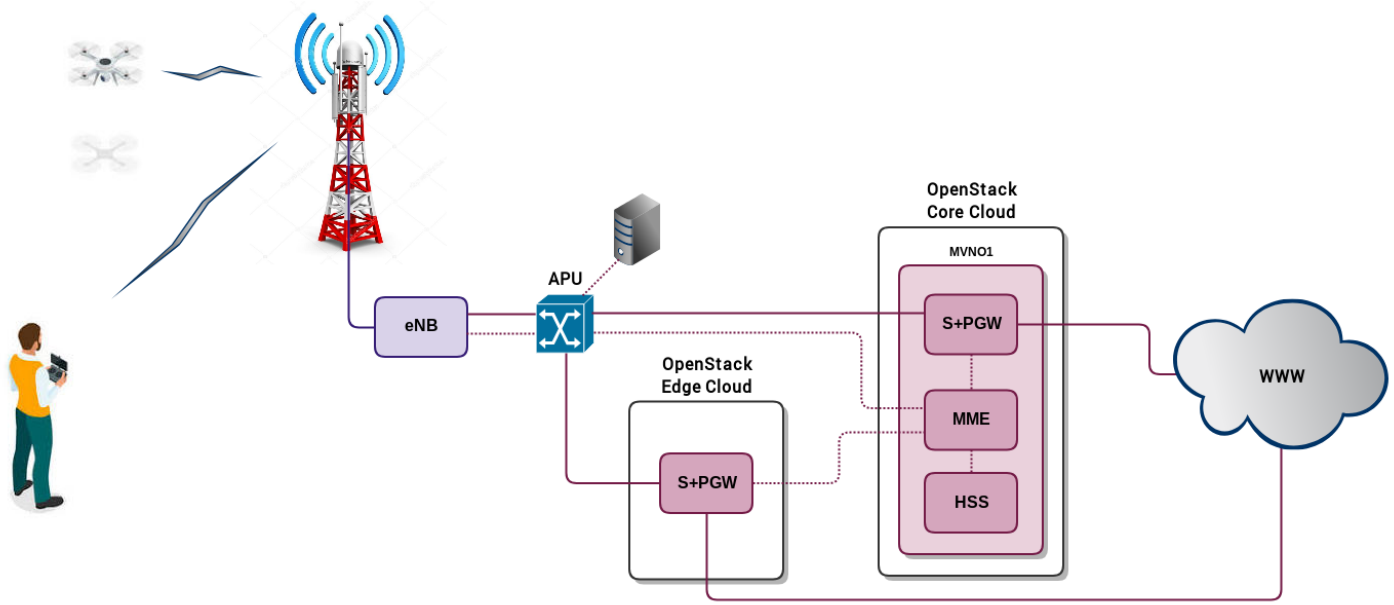
Mobile edge computing



- Components of the virtual operator that are involved in the transmission of drone control commands are “moved” to the edge.
- An OpenFlow device is used to maintain the communication.

Flexible core (V)

Mobile edge computing. Alternative I: Replicate S+PGW

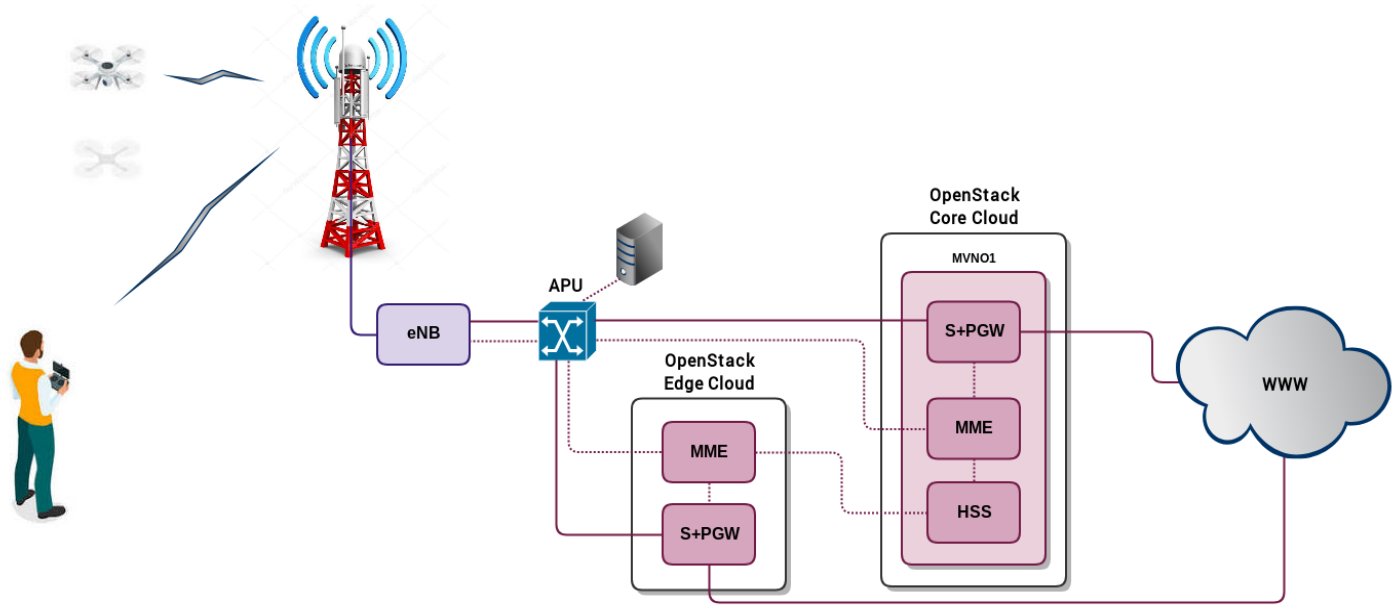


Alternative I: Replicate S+PGW

- Advantage: Only one VM has to be replicated.
- Disadvantage: Control communications still have to pass through the core.

Flexible core (VI)

Mobile edge computing. Alternative II: Replicate S+PGW and MME



Alternative II: Replicate S+PGW and MME

- Advantage: Control communication between S+PGW and MME are performed inside the edge cloud.
- Disadvantage: Two modules have to be replicated.

Conclusion

OpenAirInterface is the cornerstone for the development of new [mobile] communication technologies.

We are specially interested in the Core Network but,

- Using all the components from OAI we are able to build fully operational testbeds.
- 5G and post-5G networks require a “cross-layer” approach to meet applications’ requirements.

We are working towards the creation of a infrastructure for “Virtual Operators”

We are looking to collaborate with you in this field!!!

atlanTTic
research center
for Telecommunication Technologies

Thanks!

Felipe Gil-Castiñeira (felipe@uvigo.es)

XUNTA DE GALICIA

UNIÓN EUROPEA

Universidad de Vigo

galicia