



# Bridging ns-3 and O-RAN: a tutorial on ns-O-RAN

A. Lacava, M. Polese

Institute for the Wireless Internet of Things

Northeastern University, Boston, MA | Sapienza, University of Rome

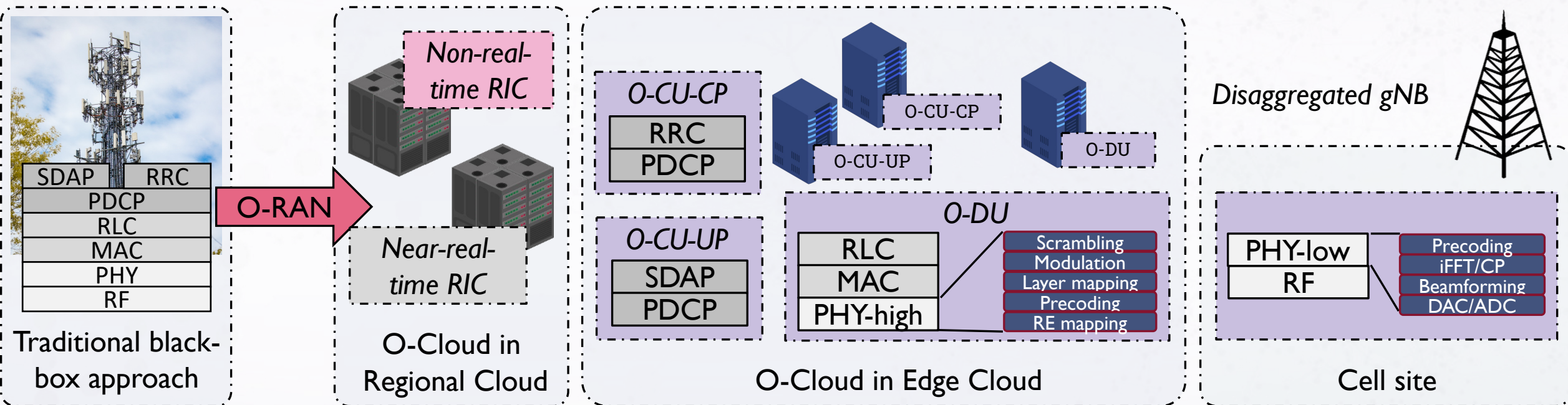
[lacava.a@northeastern.edu](mailto:lacava.a@northeastern.edu)

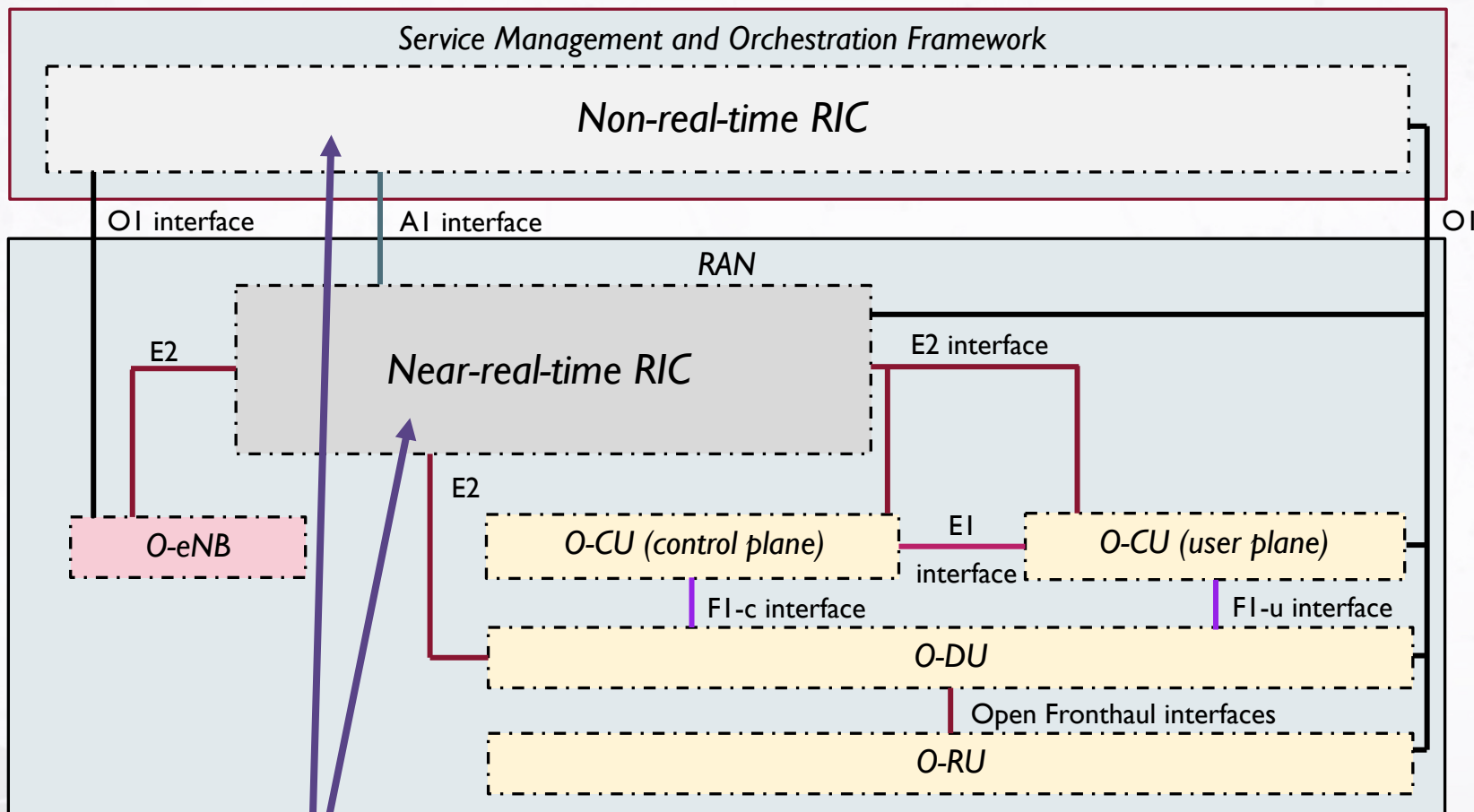
# Agenda

---

1. O-RAN – a primer
2. RIC Setup
3. ns-O-RAN setup
4. KPI monitor Setup
5. RC Control xApp
6. Scenario Zero

# Open RAN





1. Open, standardized interfaces
2. Disaggregated RAN
3. Softwarization

## 4. RAN Intelligent Controllers



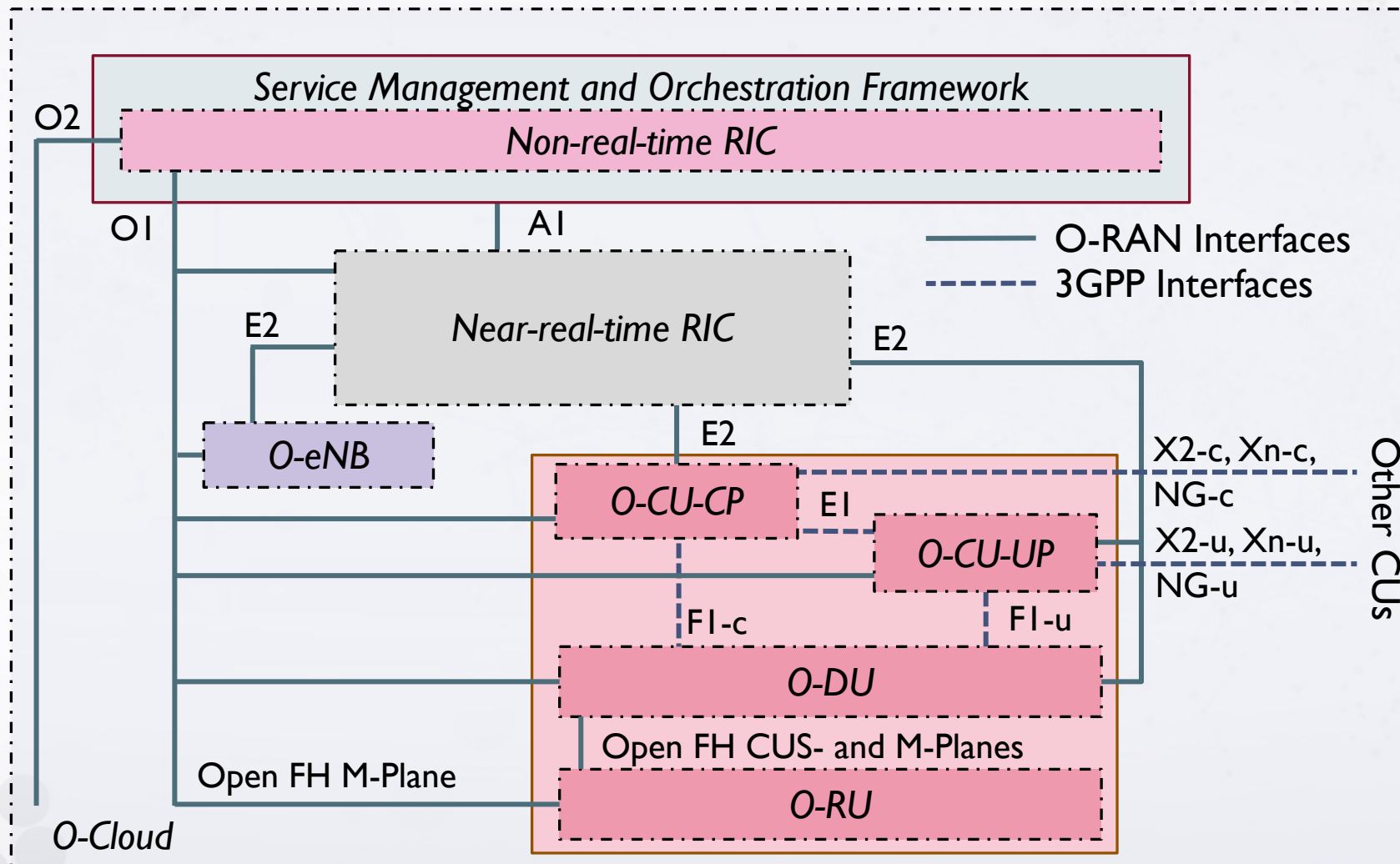
# Intelligent Control Loops

Currently supported by O-RAN

Control and learning objective	Scale	Input data	Timescale	Architecture
Policies, models, slicing	> 1000 devices	Infrastructure-level KPIs	Non real-time > 1 s	
User Session Management e.g., load balancing, handover	> 100 devices	CU-level KPIs e.g., number of sessions, PDCP traffic	Near real-time 10-1000 ms	
Medium Access Management e.g., scheduling policy, RAN slicing	> 100 devices	MAC-level KPIs e.g., PRB utilization, buffering	Near real-time 10-1000 ms	
Radio Management e.g., resource scheduling, beamforming	~10 devices	MAC/PHY-level KPIs e.g., PRB utilization, channel estimation	Real-time < 10 ms	
Device DL/UL Management e.g., modulation, interference, blockage detection	1 device	I/Q samples	Real-time < 1 ms	

For further study or not supported

# Logical architecture overview



# O-RAN Virtualization

---

## O-Cloud:

- Set of computing resources and virtualization infrastructure
  - Pooled together in one or multiple physical datacenters
- Virtualization paradigm for O-RAN
  - Decoupling between hardware and software components
  - Standardization of the hardware capabilities for the O-RAN infrastructure
  - Sharing of the hardware among different tenants
  - Automated deployment and instantiation of RAN functionalities



# O-RAN Virtualization

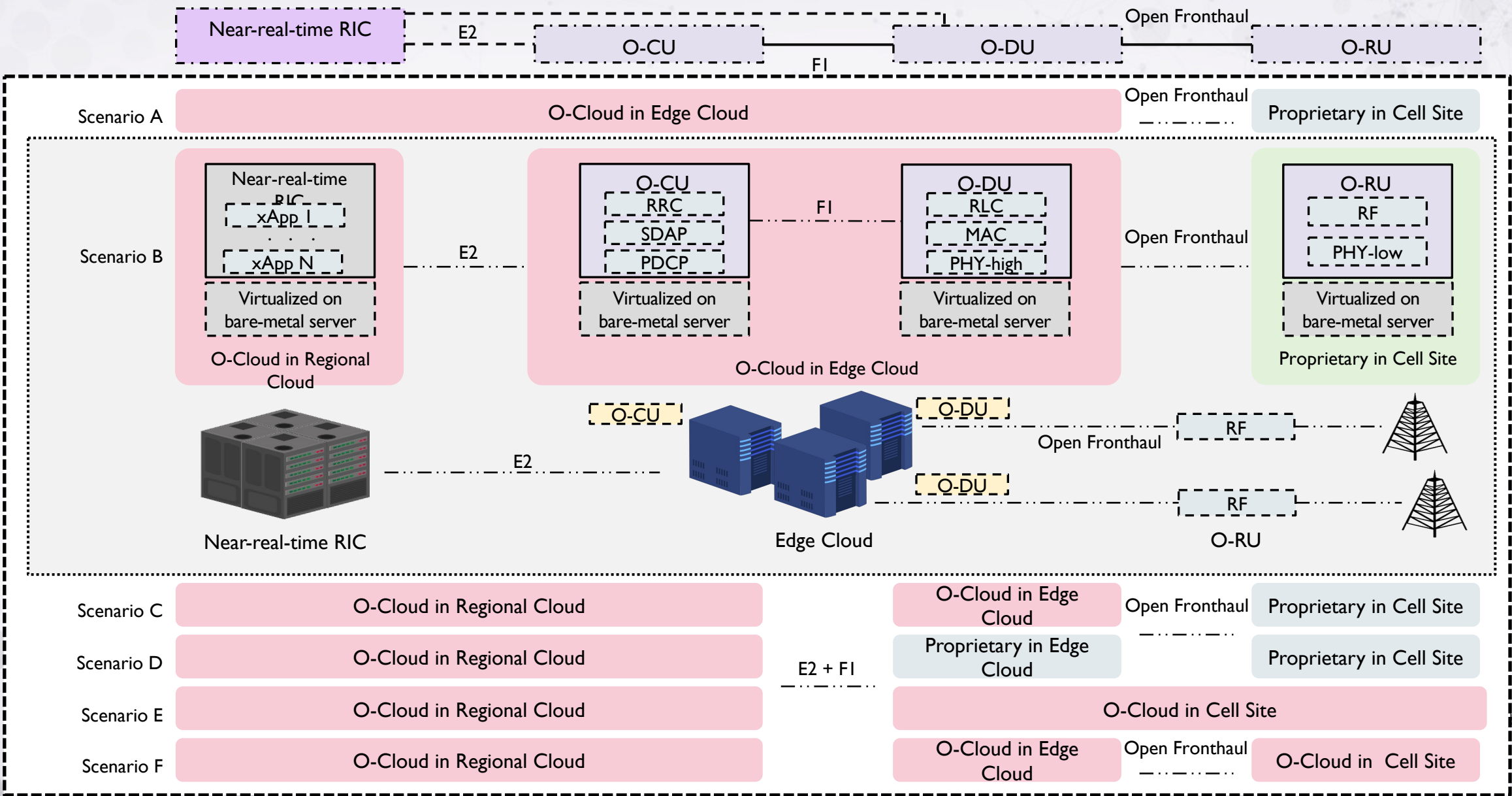
---

## Acceleration Abstraction Layers (AALs):

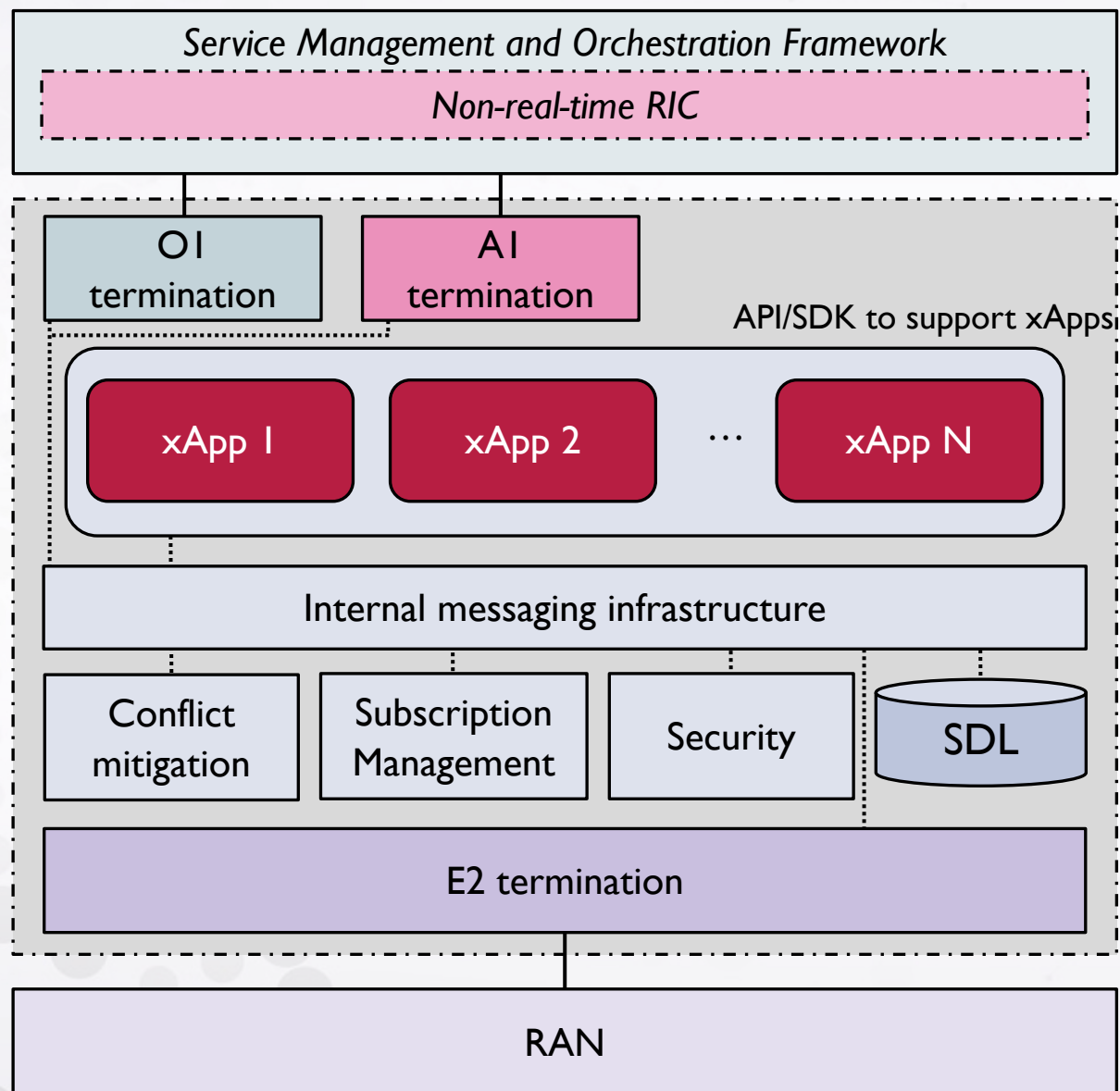
- APIs between dedicated hardware-based logical processors and the O-RAN softwarized infrastructure
  - e.g., for channel coding/FEC
- Open new opportunities for compute in the RAN
  - e.g., integrate open, programmable GPUs and FPGAs



# O-RAN deployment options

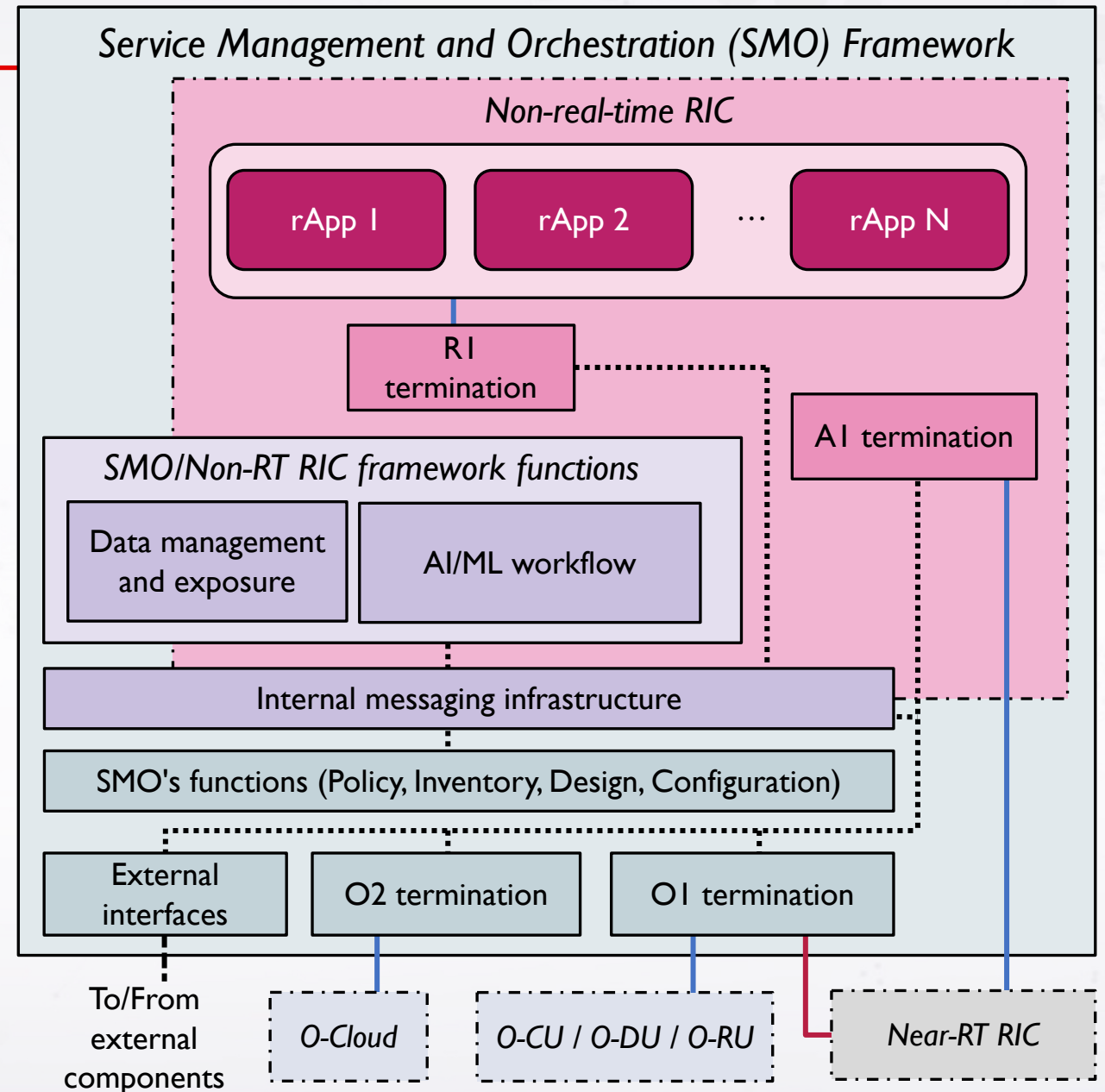
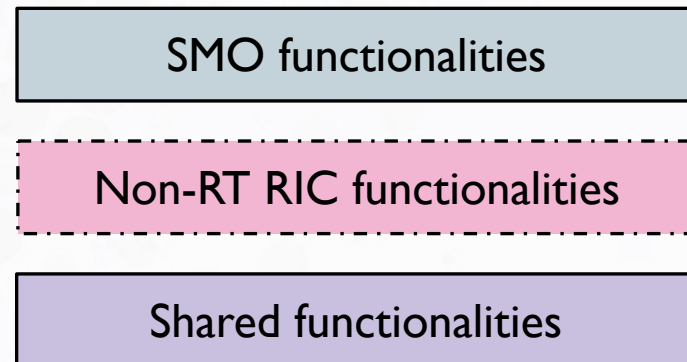


# Near-real-time RIC



- Standardized blocks and functionality
- Different implementations

# Non-real-time RIC and SMO



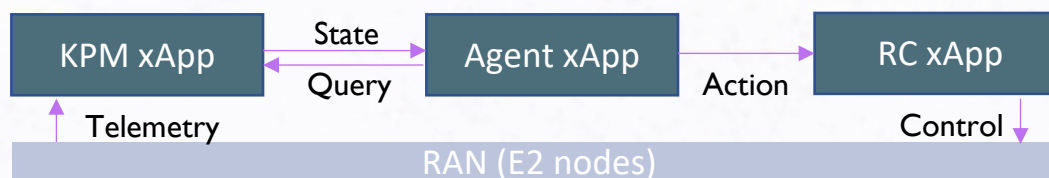
# Intelligent Use Cases



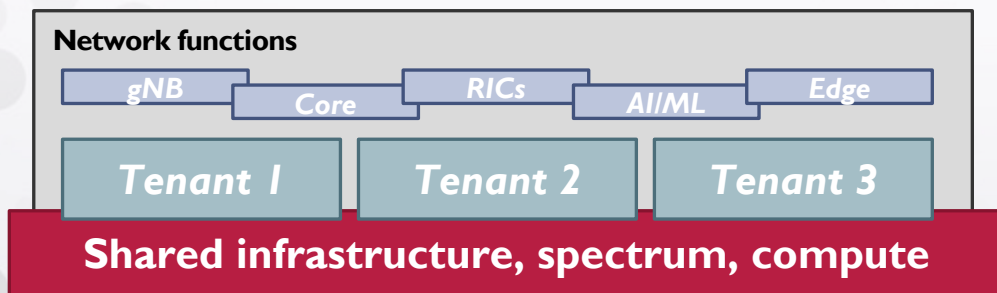
## Network slicing and scheduling



## Traffic steering



## Spectrum sharing



## Energy efficiency





# Open Challenges toward Intelligent Open RAN

---



Datasets, platforms, development and testing



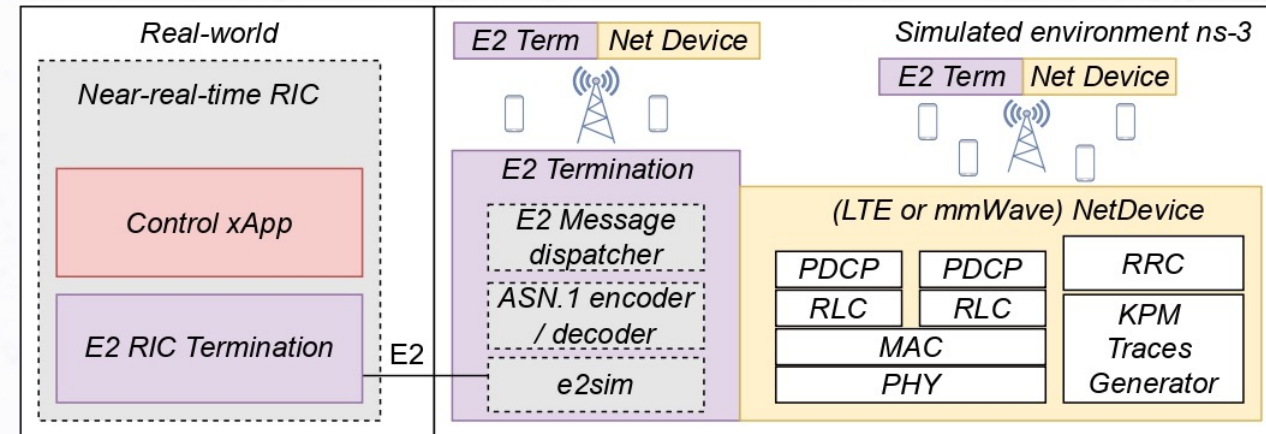
AI/ML that generalizes to different deployments and scenarios



Agile spectrum, infrastructure, and AI management

# ns-O-RAN: Simulating O-RAN 5G Systems in ns-3

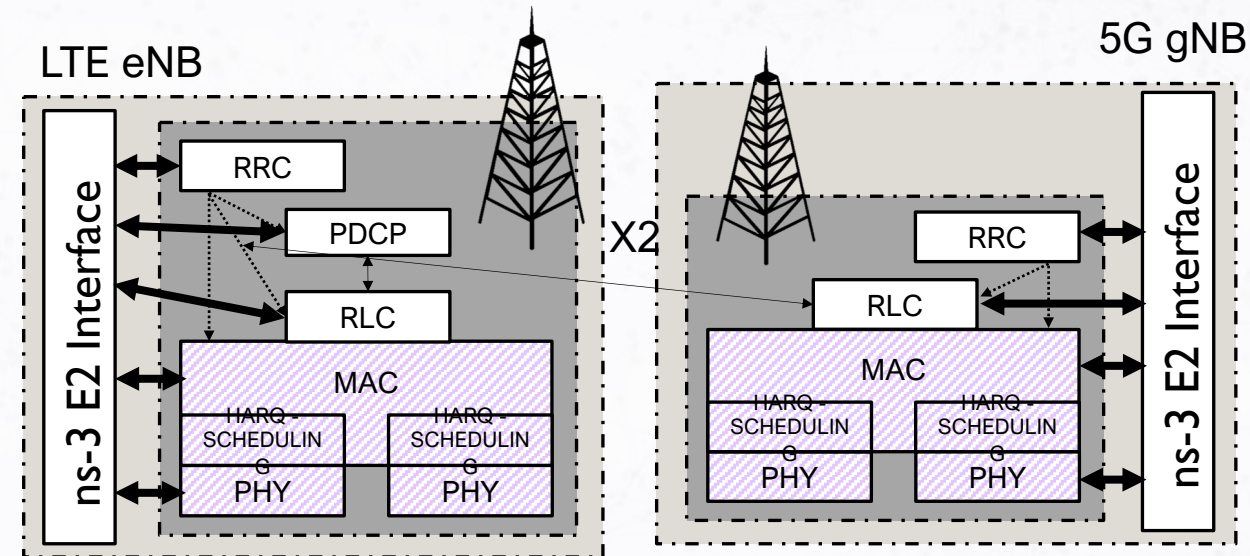
- Integration of a real-world RIC with a simulated RAN in ns-3
  - Enabling large scale simulations for O-RAN
  - KPI and Control messages exchange supported
  - Realistic dataset generation
- No infrastructure expenses
  - Highly customizable
  - Implement custom use cases
- O-RAN compliant
  - Create the xApp on ns-3 and use it on a real RAN with no software changes



*More on the implementation and architecture in the paper!*

# Enablers thanks to ns-O-RAN

- Playground for xApps
  - Test your code in a safe environment
- Environment suitable for AI
  - Define and apply the control
- Big Data collection framework
  - Stand alone mode
  - Extract context from simulated data and then adapt
  - Usable with SEM
- More on Wednesday...

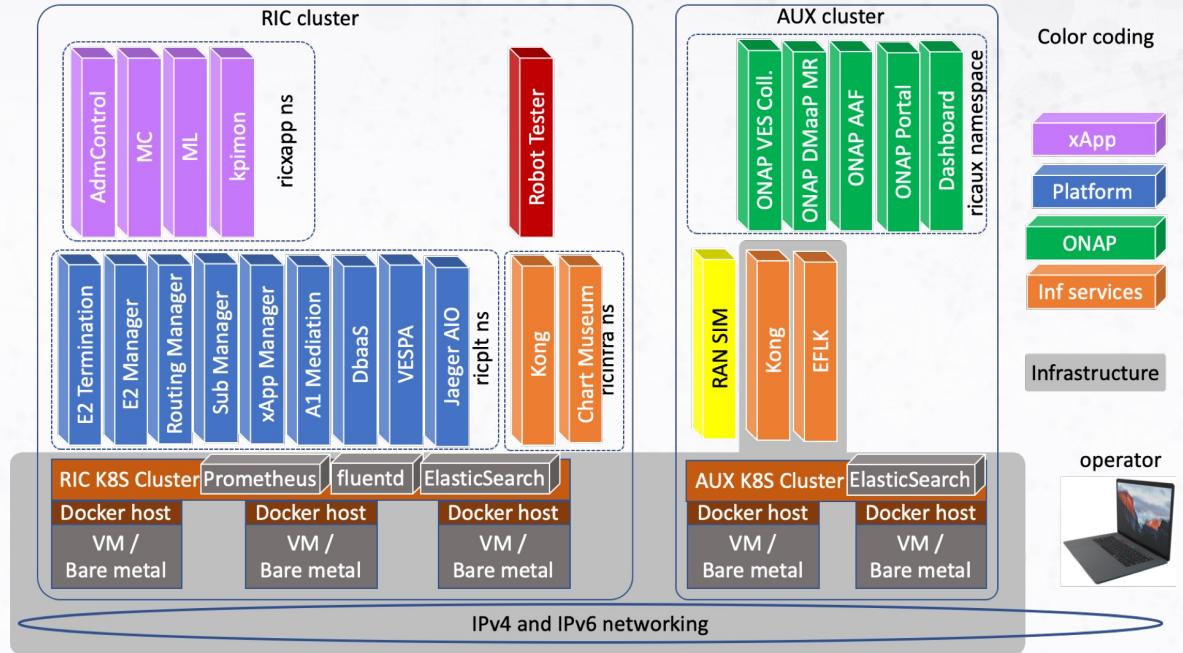


# ns-3 simulations with the OSC RIC – tutorial



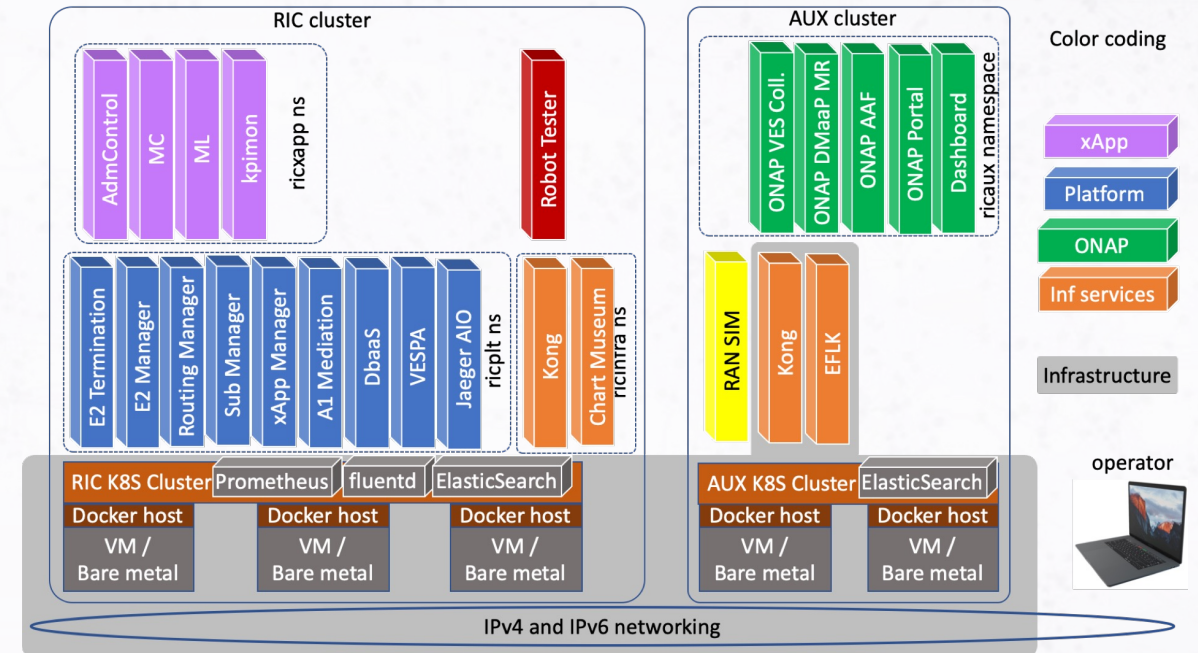
# RAN Intelligent Controller (RIC) – OSC Implementation

- Cluster of network functions at the RAN edge or in the cloud
  - Open Specifications
  - Different implementations available
    - OSC - Kubernetes
    - ColoRAN [1]
    - FlexRIC
- Functionalities implemented as microservices (*pods*)
- Namespaces* isolate services according to their role
  - ricxapp: xApps
  - ricinfra: functional pods for Kubernetes and the RIC
  - ricplt: RIC components connecting to the RAN



# RIC Major Components

- E2 Termination
  - Connection with the RAN
- E2 Manager
  - RAN Subscription Manager
- Routing Manager
  - Intra xApps and RAN
- xApp Manager
  - Handles the onboarding of the xApps
- xApps
  - Network operator applications
  - Custom control logic



# RIC Setup

- We install the E - Release
  - <https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-platform>
- Prerequisites:
  - Kubernetes
  - Docker

```
git clone https://gerrit.o-ran-sc.org/r/ric-plt/ric-dep -b e-release
cd ric-dep/
git submodule update --init --recursive --remote
./uninstall # remove old any version
./install -f ../RECIPE_EXAMPLE/example_recipe_oran_e_release.yaml
```

```
root@wines-PowerEdge-R340:/home/wines/ric-dep# kubectl get pods -n ricplt
NAME                                                    READY   STATUS    RESTARTS   AGE
deployment-ricplt-a1mediator-6ccd8896d7-ddqj4          1/1     Running   0           51d
deployment-ricplt-alarmmanager-56d79dc55-5xvqr        1/1     Running   0           51d
deployment-ricplt-appmgr-8f7467877-8k2dc              1/1     Running   0           51d
deployment-ricplt-e2mgr-66cdc4d6b6-l2hvr              1/1     Running   2           51d
deployment-ricplt-e2term-alpha-84d4db76d6-kq2zp       1/1     Running   0           38d
deployment-ricplt-o1mediator-677ff764d7-492g8         1/1     Running   0           51d
deployment-ricplt-rtmgr-578c64f5cf-mqdwg              1/1     Running   1           51d
deployment-ricplt-submgr-7f6499555d-4zp24             1/1     Running   0           51d
deployment-ricplt-vespamgr-84f7d87dfb-5zlzr           1/1     Running   0           51d
r4-infrastructure-kong-7995f4679b-v5pg9               2/2     Running   1           51d
r4-infrastructure-prometheus-alertmanager-5798b78f48-46hgt 2/2     Running   0           51d
r4-infrastructure-prometheus-server-c8ddcdf5-4mxxj    1/1     Running   0           51d
statefulset-ricplt-dbaas-server-0                    1/1     Running   0           51d
root@wines-PowerEdge-R340:/home/wines/ric-dep#
```



# ns-O-RAN Setup

- ns-3 will run in a Docker pod in the Kubernetes cluster
- Installation toolchain
  - <https://openrangym.com/tutorials/ns-o-ran>
- Dockerfile:
  - <https://github.com/wineslab/colosseum-near-rt-ric/blob/ns-o-ran/Dockerfile>
- 3 main components will be installed
  - E2sim software
  - ns-O-RAN external module
  - ns-3 MmWave module

```
docker build -t ns3 -f Dockerfile --build-arg log_level_e2sim=2 . --no-cache
```

Log Level e2Sim	Value	Description
LOG_LEVEL_UNCOND	0	Show only the unconditional logs.
LOG_LEVEL_ERROR	1	Show all the previous logs plus failures on the e2Sim side (such as errors on encoding)
LOG_LEVEL_INFO	2 (default)	Show all the previous logs plus the some info about the size of the messages
LOG_LEVEL_DEBUG	3	Show all the possible logs including the xer_printing of the ASN1.C messages



# ns-O-RAN Codebase structure

- 3 different repositories

RAN functional simulator, fork of  
<https://github.com/nyuwireless-unipd/ns3-mmwave>  
(aligned to latest updates)

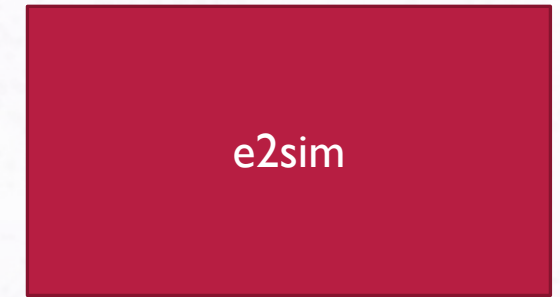


Provides RAN simulation with E2AP  
and E2SM APIs



Contributed to OSC (Oct 2022)  
<https://github.com/o-ran-sc/sim-ns3-o-ran-e2>

Fork of <https://github.com/o-ran-sc/sim-e2-interface> in Dec. 2020 – commit a8f2a



Uses e2sim as a library

# ns-O-RAN Dockerfile

```
FROM wineslab/o-ran-sc-bldr-ubuntu18-c-go:9-18.04 as buildenv
ARG log_level_e2sim=2

# Install E2sim
RUN mkdir -p /workspace
RUN apt-get update && apt-get install -y build-essential git cmake libsctp-dev autoconf automake libtool bison flex libboost-all-dev

WORKDIR /workspace

RUN git clone -b develop https://github.com/wineslab/ns-o-ran-e2-sim /workspace/e2sim

RUN mkdir /workspace/e2sim/e2sim/build
WORKDIR /workspace/e2sim/e2sim/build
RUN cmake .. -DDEV_PKG=1 -DLOG_LEVEL=${log_level_e2sim}

RUN make package
RUN echo "Going to install e2sim-dev"
RUN dpkg --install ./e2sim-dev_1.0.0_amd64.deb
RUN ldconfig

WORKDIR /workspace

# Install ns-3
RUN apt-get install -y g++ python3

RUN git clone -b release https://github.com/wineslab/ns-o-ran-ns3-mmwave /workspace/ns3-mmwave-oran
RUN git clone -b master https://github.com/o-ran-sc/sim-ns3-o-ran-e2 /workspace/ns3-mmwave-oran/contrib/oran-interface

WORKDIR /workspace/ns3-mmwave-oran

RUN ./waf configure --enable-tests --enable-examples
RUN ./waf build

WORKDIR /workspace

CMD [ "/bin/sh" ]
```

# ns-O-RAN onboarding

- Add the Docker container in the RIC
  - i.e., we create a new pod in the cluster
- We use a k8s file to ease the job
  - Yaml format to dynamically create the pod

```
apiVersion: v1
kind: Pod
metadata:
  name: ns-3-pod
  namespace: ricplt
spec:
  containers:
  - name: ns-3-pod
    image: ns3
    imagePullPolicy: Never
    command: ["sleep", "infinity"]
```

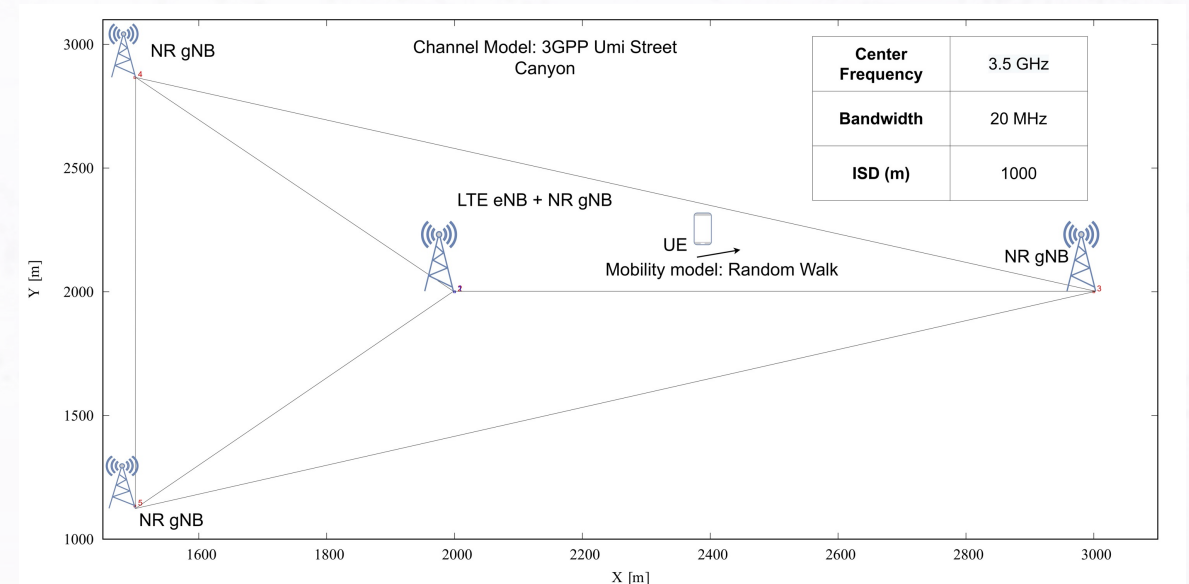
```
kubectl apply -f ns-o-ran-pod.yaml
```

```
wines@wines-PowerEdge-R340:~$ sudo kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
ns-3-pod      1/1     Running   0           79d
```

# Test ns-O-RAN in stand-alone mode

```
./ns3 run "scratch/scenario-zero.cc --simTime=10 --enableE2FileLogging=1"
```

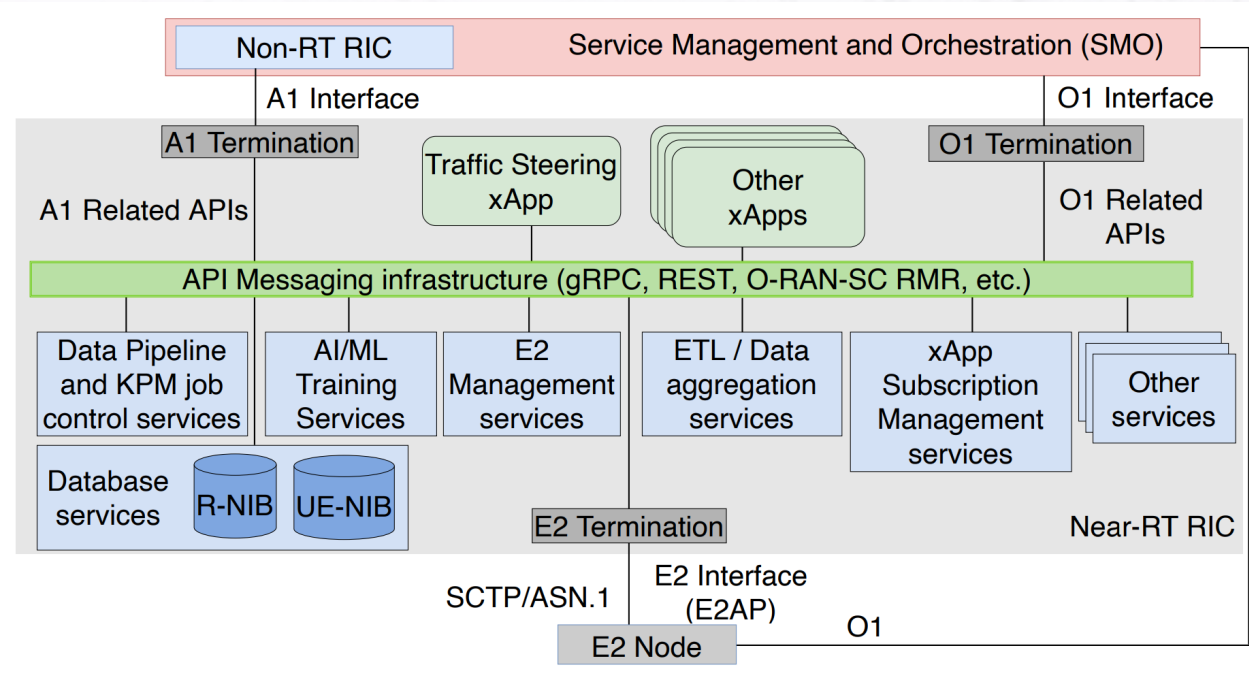
- ns-3 without RIC
  - Save logs and RAN telemetry
- Scenario Zero
  - 1 eNB, 4 gNB
  - 12 UEs
  - simTime: seconds of the simulation
  - enableE2FileLogging: if true, ns-O-RAN is in stand alone mode
  - e2TermIp: IP address of the RIC E2 Termination





# Connecting the xApps

- Working with xApps is **hard**:
  - Compatibility among versions
  - Handling of Subscription IDs
  - Internal routing of the E2 Messages
- The flexibility of ns-O-RAN can help



# Onboarding a complete xApp

- xApps are designed to be plug and play
- Once you have configurations details they can be deployed with a zero-touch approach
- Two major files are needed to load the xApp in the RIC:
  - config-file.json
  - schema.json
- The result is the xApp live and a docker container

```
# Start chartmuseum
docker run --rm -u 0 -it -d -p 8090:8080 -e DEBUG=1 -e STORAGE=local -e
STORAGE_LOCAL_ROOTDIR=/charts -v $(pwd)/charts:/charts chartmuseum/chartmuseum:latest

# setting environment variable
export CHART_REPO_URL=http://0.0.0.0:8090
# onboard xApp
dms_cli onboard config-file.json schema.json

# install xApp
dms_cli install {name_xapp} {version_xapp} ricxapp
# uninstall xApp
dms_cli uninstall {name_xapp} ricxapp[]
```



# Customizing the xApp

---

- Changing the code requires recreating the container:
  - Entrypoint
  - Source code
- After building the image, it should be pushed to a registry because dms cli
- Modify then the config-file.json to point the correct registry and image
- To manually work in the container the Dockerfile shall have as ENTRYPOINT the command: ['sleep', 'infinity']

```
# start registry on localhost
docker run -d -p 5000:5000 -- name registry registry:2

# build the docker image always tagging it with -t as 127.0.0.1:5000/${name_xapp}:${version}

# push the image to the registry
docker push 127.0.0.1:5000/${name_xapp}:${version}
```



# Before moving on – ASN.1 Definitions

---

- Encoding technique used by cellular networks
  - Hard to customize
  - Follows standard definitions
  - Very efficient
  - Must be consistent
- For this tutorial we use:
  - E2AP ASN from OSC G release
  - Custom ns-O-RAN E2SM KPI
  - E2SM RC from G release
- All of them can be found at:
  - <https://github.com/wineslab/libe2proto/tree/ns-o-ran>

```
RANfunction-Name ::= SEQUENCE{
    ranFunction-ShortName   PrintableString (SIZE(1..150,...)),
    ranFunction-E2SM-0ID    PrintableString (SIZE(1..150,...)),
    ranFunction-Description PrintableString (SIZE(1..150,...)),
    ranFunction-Instance    INTEGER OPTIONAL,
    ...
}

RIC-EventTriggerStyle-Item ::= SEQUENCE{
    ric-EventTriggerStyle-Type    RIC-Style-Type,
    ric-EventTriggerStyle-Name    RIC-Style-Name,
    ric-EventTriggerFormat-Type   RIC-Format-Type,
    ...
}
```

All the xApps are taken from the  
OSC repositories



# xApp KPI monitor

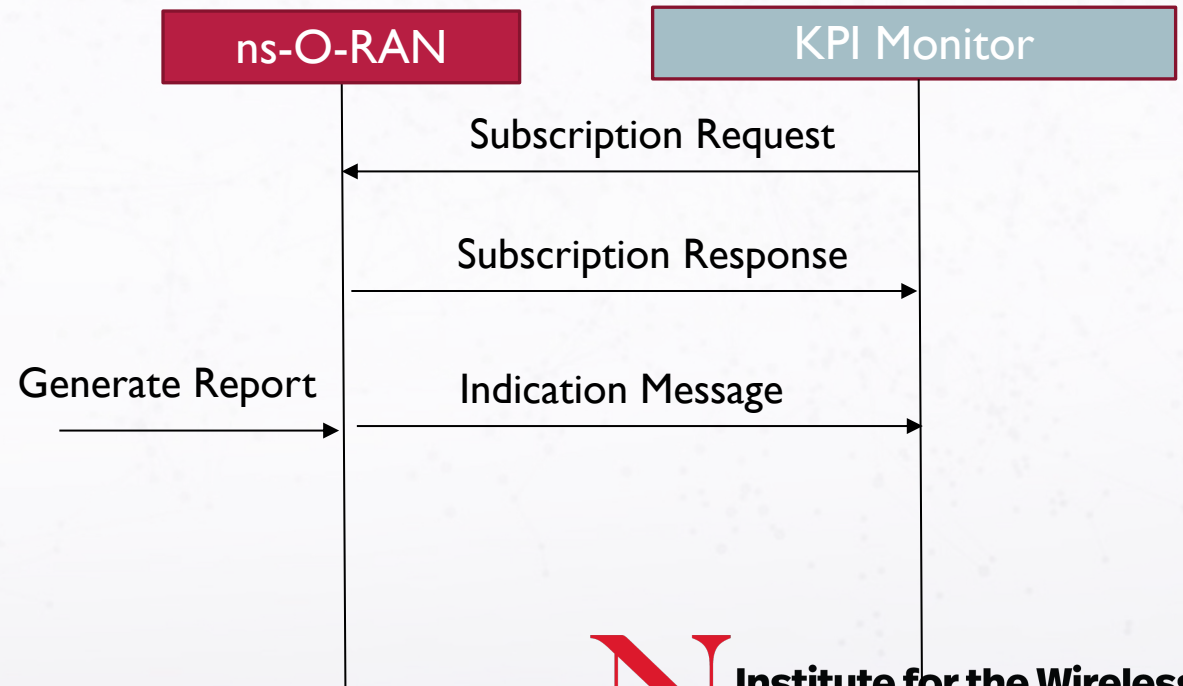
- Monitoring xApp

- Send the E2 Subscription Request
- Receives the Indication Messages from the RAN
- Decode the parameters
- Store the values in the real time database (not in this demo)

- Available here:

- <https://github.com/wineslab/ns-o-ran-scp-ric-app-kpimon>

```
git clone https://github.com/wineslab/ns-o-ran-scp-ric-app-kpimon kpimon -b libe2proto
cd kpimon
./launch_app.sh
# a shell should spawn inside the kpimon pod
# Actually launch the routine of the xApp
/kpimon -f /opt/ric/config/config-file.json
```



# xApp KPI monitor – launch script

---

- Creates an env. variable to specify the k8s backend.
- Uninstall old versions
- Building the xApp from source
- Push the image to the registry
- Onboards the xApp, using the descriptor and validation schema.
- Installing the xApp in the RIC
- After 10s, the script returns the name of the pod in the *ricxapp* namespace and the command to shell inside it

```
#!/bin/bash
#set -x

export CHART_REPO_URL=http://0.0.0.0:8090

dms_cli uninstall xappkpimon ricxapp

docker build . -f Dockerfile -t 127.0.0.1:5000/kpimon_master:1.0.0 # --no-cache

docker push 127.0.0.1:5000/kpimon_master:1.0.0

# dms_cli onboard config.json schema.json

dms_cli install xappkpimon 1.0.0 ricxapp

echo "Wait for 10 seconds"
sleep 10

unset $pod_name

pod_name=$(kubectl get pods -n ricxapp --no-headers -o custom-columns=":metadata.name")

echo kubectl exec -ti -n ricxapp $pod_name bash

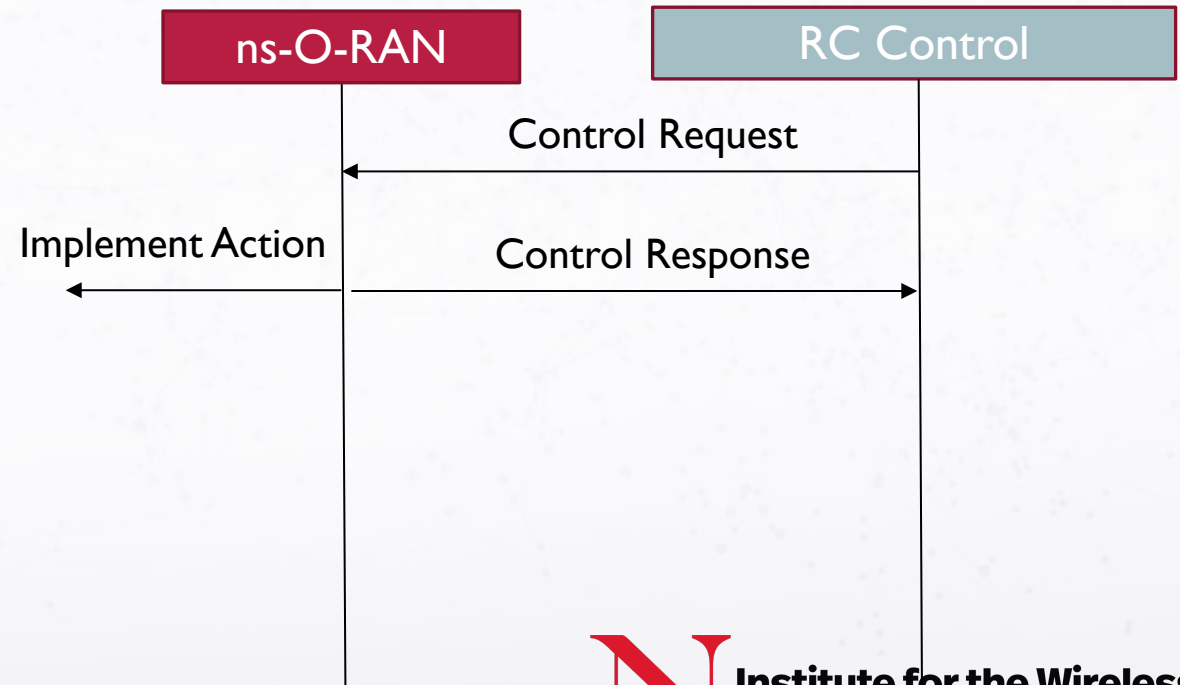
# To run the kpimon
# ./kpimon -f /opt/ric/config/config-file.json
```

# RC Control xApp

- Implements the TS use case
  - Send the E2 RC Control Action to the RAN
    - For example, an handover command
  - Can be used as a network function by other xApps
  - Server GRPC that create the controls on demand
- Available here:
  - <https://github.com/wineslab/ns-o-ran-xapp-rc>



```
git clone https://github.com/wineslab/ns-o-ran-xapp-rc xapp-rc
cd xapp-rc
./launch_app.sh
```



# RC Control Command

---

- GRPC commands can be executed:
  - by xApps
  - manually with grpcurl:
    - <https://github.com/fullstorydev/grpcurl>

```
RC_XAPP_IP=10.104.101.157
```

```
grpcurl -plaintext -d "{ \"e2NodeID\": \"360000000\", \"plmnID\": \"313131\", \"ranName\": \"gnb_131_133_310000000\",  
  \"RICE2APHeaderData\": { \"RanFuncId\": 300, \"RICRequestorID\": 2 }, \"RICControlHeaderData\": { \"ControlStyle\": 3,  
  \"ControlActionId\": 1, \"UEID\": \"00003\" }, \"RICControlMessageData\": { \"RICControlCellTypeVal\": 4, \"TargetCellID\": \"11103\"  
  }, \"RICControlAckReqVal\": 0 }" ${RC_XAPP_IP}:7777 rc.MsgComm.SendRICControlReqServiceGrp
```



# Combining things together

---

- We get the IP of the E2 termination
- We start the simulation
- Start and observe the monitoring of the kpimon
- Send control action to ns-O-RAN with GRPC

```
kubectl get pods -n ricplt -o wide
```

```
NS_LOG="RicControlMessage" ./ns3 run "scratch/scenario-zero.cc --simTime=15 --e2TermIp=10.244.0.161"
```

# Useful commands for working in the RIC

---



```
# check IP of RIC components (especially) E2 Termination
```

```
kubectl get pods -A -o wide -n ricplt
```

```
# check IP of the services in the xApp namespace
```

```
kubectl get svc -o wide -n ricxapp
```

# Useful commands when working with the RIC

---

- APP manager:
  - View all the deployed xApps
  - Manually remove an xApp
- Subscription Manager:
  - View subscription IDs

```
curl -X GET -H "Content-Type:application/json" http://$APPMGR:8080/ric/v1/xapps | jq .
```

```
curl -X POST -H "Content-Type: application/json" \
http://$APPMGR:8080/ric/v1/deregister -d \
'{"appInstanceName": "cnn-31", "appName": "cnn-31"}'
```

```
curl -X GET "http://${SUBMGR}:8088/ric/v1/subscriptions"
```

# Useful commands for working in the RIC

---

- Routing Manager:
  - View routes
- Manually add a route
- Manually delete a route

```
curl -X GET -H "accept: application/json" \ "http://${RTMGR}:3800/ric/v1/getdebuginfo" | jq .
```

```
curl -X 'POST' \
  "http://${RTMGR}:3800/ric/v1/handles/addrmroute" \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "TargetEndPoint": "service-ricxapp-cnn-1-rmr.ricxapp:4560",
      "MessageType": 12050,
      "SenderEndPoint": "",
      "SubscriptionID": 1
    }
  ]'
```

```
curl -X 'DELETE' \
  "http://${RTMGR}:3800/ric/v1/handles/delrmroute" \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "TargetEndPoint": "service-ricxapp-cnn-1-rmr.ricxapp:4560",
      "MessageType": 12050,
      "SenderEndPoint": "",
      "SubscriptionID": 1
    }
  ]'
```



# Thanks for the attention!

## Questions?

<https://openrangym.com/tutorials/ns-o-ran>

[1] Lacava, Andrea, Michele Polese, Rajarajan Sivaraj, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. "Programmable and customized intelligence for traffic steering in 5g networks using open ran architectures." IEEE Transactions on Mobile Computing (2023).

[2] Andrea Lacava, Matteo Bordin, Michele Polese, Rajarajan Sivaraj, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. 2023. Ns-O-RAN: Simulating O-RAN 5G Systems in ns-3. In Proceedings of the 2023 Workshop on ns-3 (WNS3 '23). Association for Computing Machinery, New York, NY, USA, 35–44. <https://doi.org/10.1145/3592149.3592161>