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

A. Lacava, S. Maxenti, M. Polese
Institute for the Wireless Internet of Things
Northeastern University, Boston, MA | Sapienza, University of Rome
lacava.a@northeastern.edu | andrea.lacava@uniroma1.it
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

Traditional black-box approach

Non-real-time RIC

Near-real-time RIC

O-Cloud in Regional Cloud

O-CU-CP
- RRC
- PDCP

O-CU-UP
- SDAP
- PDCP

O-DU
- RLC
- MAC
- PHY-high

Disaggregated gNB

PHY-low
- RF

Precoding
- iFFT/CP
- Beamforming
- DAC/ADC

Institute for the Wireless Internet of Things at Northeastern
- a reference architecture for programmable NextG

Service Management and Orchestration Framework

Non-real-time RIC

Near-real-time RIC

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

4. RAN Intelligent Controllers
## Intelligent Control Loops

<table>
<thead>
<tr>
<th>Control and learning objective</th>
<th>Scale</th>
<th>Input data</th>
<th>Timescale</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policies, models, slicing</td>
<td>&gt; 1000 devices</td>
<td>Infrastructure-level KPIs</td>
<td>Non real-time &gt; 1 s</td>
<td>Service Management and Orchestration (SMO)</td>
</tr>
<tr>
<td>User Session Management</td>
<td>&gt; 100 devices</td>
<td>CU-level KPIs</td>
<td>Near real-time 10-1000 ms</td>
<td>non real-time RIC</td>
</tr>
<tr>
<td>Medium Access Management</td>
<td>&gt; 100 devices</td>
<td>MAC-level KPIs</td>
<td>Near real-time 10-1000 ms</td>
<td></td>
</tr>
<tr>
<td>Radio Management</td>
<td>~10 devices</td>
<td>MAC/PHY-level KPIs</td>
<td>Real-time &lt; 10 ms</td>
<td></td>
</tr>
<tr>
<td>Device DL/UL Management</td>
<td>1 device</td>
<td>I/Q samples</td>
<td>Real-time &lt; 1 ms</td>
<td></td>
</tr>
</tbody>
</table>

- **Non real-time RIC**: > 1 s
- **Near real-time RIC**: 10-1000 ms
- **Real-time RIC**: < 1 ms

*Currently supported by O-RAN*

*For further study or not supported*
Logical architecture overview

Service Management and Orchestration Framework

Non-real-time RIC

Near-real-time RIC

O1

E2

E2

E2

O-RAN Interfaces

3GPP Interfaces

O-Cloud

O-eNB

Open FH M-Plane

O-CU-CP

O-CU-UP

O-DU

Open FH CUS- and M-Planes

O-RU

X2-c, Xn-c, NG-c

X2-u, Xn-u, NG-u

Other CUs
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

Scenario A
- O-Cloud in Edge Cloud
- Proprietary in Cell Site

Scenario B
- O-Cloud in Regional Cloud
- Virtualized on bare-metal server
- Near-time RIC

Scenario C
- O-Cloud in Regional Cloud

Scenario D
- O-Cloud in Regional Cloud

Scenario E
- O-Cloud in Regional Cloud

Scenario F
- O-Cloud in Regional Cloud
Near-real-time RIC

- Standardized blocks and functionality
- Different implementations
Non-real-time RIC and SMO

Service Management and Orchestration (SMO) Framework

Non-real-time RIC

SMO functionalities

Non-RT RIC functionalities

Shared functionalities

SMO/Non-RT RIC framework functions

Data management and exposure

AI/ML workflow

Internal messaging infrastructure

SMO’s functions (Policy, Inventory, Design, Configuration)

External interfaces

O2 termination

O1 termination

To/From external components

0-Cloud

0-CU / 0-DU / 0-RU

Near-RT RIC

AI termination

RI termination
Intelligent Use Cases

- Network slicing and scheduling
  - eMBB PRBs + scheduling policy
  - MTC PRBs + scheduling policy
  - URLLC PRBs + scheduling policy

- Spectrum sharing

- Traffic steering
  - KPM xApp
  - Agent xApp
  - RC xApp
  - Telemetry
  - Control

- Energy efficiency
  - Compute
  - Scaling
  - Near RT-RIC

Shared infrastructure, spectrum, compute
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 tomorrow!
ns-3 simulations with an O-RAN 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 G-Release
  - [https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-platform](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-platform)

- Prerequisites:
  - Kubernetes
  - Docker

```
root@wines-PowerEdge-R340:/home/wines/ric-dep# kubectl get pods -n ricpltl
NAME                      READY   STATUS    RESTARTS   AGE
deployment-ricpltl-amediator-6ccd8896d7-ddqj4  1/1       Running   0       51d
deployment-ricpltl-alarmmanager-56d79dc55-5xvqr  1/1       Running   0       51d
deployment-ricpltl-appmgr-8f7467877-8k2dc       1/1       Running   0       51d
deployment-ricpltl-e2mgr-66c6d4d6b6-12hvr        1/1       Running   2       51d
deployment-ricpltl-e2term-alpha-84d4db76d6-kq2zp  1/1       Running   0       38d
deployment-ricpltl-omediator-677ff764d7-492g8     1/1       Running   0       38d
deployment-ricpltl-rtmgr-578c6f5fcf-mdwng        1/1       Running   1       51d
deployment-ricpltl-submgr-7f6499555d-4zp24        1/1       Running   0       51d
deployment-ricpltl-vespomgr-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-c8ddcdf5-4mhj  1/1       Running   0       51d
statefulset-ricpltl-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
ns-O-RAN Codebase structure

• 3 different repositories

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

ns3-mmWave

ns-O-RAN

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

e2sim

Uses e2sim as a library
ns-O-RAN Dockerfile

FROM wineslab/o-ran-sc-bldr-ubuntu18-c-go:9-u18.04 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 libscip-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
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-mmmwave /workspace/ns3-mmmwave-oran
RUN git clone -b master https://github.com/o-ran-sc/sim-ns3-o-ran-e2 /workspace/ns3-mmmwave-oran/contrib/oran-interface

WORKDIR /workspace/ns3-mmmwave-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

```yaml
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
```

```
<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns-3-pod</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>79d</td>
</tr>
</tbody>
</table>
```
Test ns-O-RAN in stand-alone mode

```bash
./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 $USER -it -d -p 8090:8000 --rm DEBUG=1 --疮 STORAGE=local --疮 STORAGE_LOCAL_ROOTDIR=/charts --v $(pwd)/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} rlxapp
# Uninstall xApp
dms_cli uninstall {name_xapp} rlxapp]
```

On-demand

Automation

Reconfigurability
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/${name}_app:${version}
docker build -t 127.0.0.1:5000/${name}_app:${version}

# push the image to the registry
docker push 127.0.0.1:5000/${name}_app:${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

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
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 rixapp namespace and the command to shell inside it

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

export CHART_REPO_URL=http://0.0.0.0:8090
dms_cli uninstall xappkplmon rixapp
docker build . -f Dockerfile -t 127.0.0.1:5000/kplmon_master:1.0.0 # --no-cache
docker push 127.0.0.1:5000/kplmon_master:1.0.0
# dms_cli onboard config.json schema.json
dms_cli install xappkplmon 1.0.0 rixapp
echo "Wait for 10 seconds"
sleep 10
unset $pod_name
pod_name=$(kubectl get pods -n rixapp --no-headers -o custom-columns=:metadata.name)
echo kubectl exec -ti -n rixapp $pod_name bash
# To run the kplmon
# ./kplmon -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
./launch_app.sh
```
• GRPC commands can be executed:
  • by xApps
  • manually with grpcurl:
    • [https://github.com/fullstorydev/grpcurl](https://github.com/fullstorydev/grpcurl)
Putting things together

- We get the IP of the e2 termination
- We launch the simulation
- Launch and observe the monitoring of the kpimon
- Send control action to ns-O-RAN with GRPC
Useful commands for working in the RIC

```
# check IP of RIC components (especially) E2 Termination
kubectl get pods -A -o wide -n ricpltl

# check IP of the services in the xApp namespace
kubectl get svc -o wide -n ricxapp
```
Useful commands for working in the RIC

- **APP manager:**
  - View all the deployed xApps
  - Manually remove an xApp

- **Subscription Manager:**
  - View subscription IDs

```bash
```

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

```bash
curl -X GET "http://$SUBMGR:8088/v1/subscriptions"
```
Useful commands for working in the RIC

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

```bash
```

```bash
curl -X 'POST' 
 "http://$(RTMgr):3800/ric/v1/handles/addrmrroute" 
 -H 'accept: application/json' 
 -H 'Content-Type: application/json' 
 -d '[
    {
      "TargetEndPoint": "service-ricxapp-cnn-1-rmr.ricxapp:4560",
      "MessageType": 12050,
      "SenderId": "",
      "SubscriptionID": 1
    }
  ]';
```

```bash
curl -X 'DELETE' 
 "http://$(RTMgr):3800/ric/v1/handles/delrmrroute" 
 -H 'accept: application/json' 
 -H 'Content-Type: application/json' 
 -d '[
    {
      "TargetEndPoint": "service-ricxapp-cnn-1-rmr.ricxapp:4560",
      "MessageType": 12050,
      "SenderId": "",
      "SubscriptionID": 1
    }
  ]';
```
Thanks for the attention!

Questions?
