O-RAN with Machine Learning in ns-3

WNS3 23 – June 2023

Wesley Garey\textsuperscript{1}, Tanguy Ropitault\textsuperscript{2,3}, Richard Rouil\textsuperscript{1}, Evan Black\textsuperscript{1}, Weichao Gao\textsuperscript{2,4}

\textsuperscript{1} National Institute of Standards and Technology Gaithersburg, Maryland, USA
\textsuperscript{2} Associate, National Institute of Standards and Technology, Maryland, USA
\textsuperscript{3} Prometheus Computing LLC Bethesda, Maryland, USA
\textsuperscript{4} Dakota Consulting, Inc. Silver Spring, Maryland, USA
Disclaimer

Certain equipment, instruments, software, or materials are identified in this presentation in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.
Traditional Radio Access Network (RAN)

- Proprietary and highly integrated system
  ⇒ limited flexibility, scalability, and vendor lock-in

Open-RAN (O-RAN)

- Evolve the RAN to be open, intelligent, interoperable and autonomous thanks to **disaggregation, virtualization and open interfaces**

Usage of Artificial Intelligence/Machine Learning (AI/ML):
- Autonomous control using real-time data and statistics

**Evaluation: Complex**

- Understanding the interactions and the performance is a challenging task
  ⇒ **System-level simulations**
- **ns3-oran**: [https://github.com/usnistgov/ns3-oran](https://github.com/usnistgov/ns3-oran)
  - Implementation of O-RAN architecture with a focus on ease of installation and extensibility. It includes:
    - Near-Real Time (RT) RAN Intelligent Controller (RIC) and E2 interface O-RAN components
    - Fast and efficient ML
About O-RAN Architecture

Service Management and Orchestration (SMO) Framework
- Houses the non-Real Time (RT) Real-Time RIC: manages and orchestrates the top-level control of the network functions
- Ensures efficient operation through policy management and facilitates the deployment of AI/ML models

Network Functions
- Intermediary between the Non-RT RIC and the Radio Unit (RU) component
- Houses the Near-RT RIC:
  - Goal: optimizing network performance by dynamically adjusting network parameters based on real-time conditions and policies.
  - Connected to multiple components through the E2 interface to request O-RAN actions and/or provide information about network status and performance

O-Cloud
- Pool of physical servers hosting the Network Functions.
- Allows to host virtualized components
- Cloud-based: Dynamic scalability and agile deployment of services => Flexible RAN
About O-RAN
Architecture: Control Loop

Non-RT RIC Control Loop > 1 s:
Managing less time-sensitive aspects of the radio access network, e.g.,
Network-wide optimization, longer-term radio resource management,
policy creation and updates, or intelligent load balancing

Near RT RIC >= 10 ms and < 1 s:
Fine-grained control and optimization of the RAN, e.g., interference
management, handovers, load balancing, or spectrum sharing

O-DU and O-RU control loop*: < 1 ms:
Device management and optimization, e.g, scheduling, beam management

* Currently discussed
Focus on the near-RT RIC

- Collects real-time data from RAN components, performs analytics, and dynamically adjusts network parameters to align with the defined policies and guidelines, and this to improve performance
- The Non-RT RIC provides the policies
Focus on the near-RT RIC

- Collects real-time data from RAN components, performs analytics, and dynamically adjusts network parameters to align with the defined policies and guidelines, and this to improve performance

- The Non-RT RIC provides the policies

- Multiple components
Focus on the near-RT RIC

**xApp**

- **xApp**: Software applications running on the near-RT RIC designed to address specific use cases within the RAN domain (may be provided by any third party)

- **xApp**
  - DIGESTS AND REALIZES THE POLICIES PUSHED TO THE NEAR-RT RIC BY THE NON-RT RIC
  - RESPONSIBLE FOR ANALYZING DATA STORED AT THE NEAR-RT RIC TO MAKE USEFUL DECISIONS

- **xApp examples**
  - Spectrum sharing
  - Cell selection
Focus on the near-RT RIC
Database, Conflict Mitigation, and ML

- **Database**: Where all of the data is stored that may need to be accessed by the Near-RT RIC and its components, such as the xApps

- **Conflict Mitigation**
  - Filter that processes all of the actions that xApps plan to carry out in the RAN to handle direct, indirect, and implicit conflicts
  - Needed as it is possible as different xApps goals might result in actions that affect each other performance
  - Responsibility of the CM to both discover these conflicts and form resolutions

- **ML**: The ML module provides ML support so that xApps can train models and use them to make inferences
Focus on the near-RT RIC
The E2 Interface

Serves as the logical link between E2 nodes and the Near-RT RIC (E2 node is any RAN component with an E2 interface between itself and the Near-RT RIC)

Two protocols:
- *E2 Application Protocol (E2AP)*: helps E2 nodes establish a link with Near-RT RIC
- *E2 Service Model (E2SM)*: enables the E2 node to *report* network-related information to the Near-RT RIC and receive *control* and *policy* messages

Through E2AP and E2SM, E2 interface enables intelligent communication and policy enforcement in the network, enhancing the effectiveness of Near-RT RIC
The O-RAN ns-3 module
Available at: https://github.com/usnistgov/ns3 oran

- Developed with **extensibility** and **ease of use** as primary considerations

Implement abstractions of the **Near-RT RIC** and virtual **E2 interface***

Implement the **xApp**, **Database**, **ML** and **Conflict Mitigation** module

*No SMO or O-Cloud as policy-level design and RAN virtualization are out of the scope of our model.
The O-RAN ns-3 module
How does it compare to other existing ns-3 O-RAN Tools

- **OpenRAN Gym [1]: Very comprehensive O-RAN implementation**
- Requires the use of Docker containers and/or a true implementation of a Near-RT RIC, the E2 interface, and remaining O-RAN components, in combination with a specialized version of ns-3
- Attractive for vendors and operators who intend to implement and later deploy their solution

- **Simpler approach**: convenient for the researcher still in exploratory or design phase and with interest in the effect of O-RAN deployment rather than the deployment itself
- Requires only the base version of ns-3 and the module that we provide

Near-RT RIC
The O-RAN ns-3 module

Architecture

Near-RT RIC

E2 Terminator

RxReport

TxCommands

E2 Node

E2 Node Terminator

Reporter

Report Trigger

Reporter

Report Trigger

Reporter

Report Trigger

Report Trigger: Periodic or events based
The O-RAN ns-3 module
Architecture

Near-RT RIC

Data repository is the access point to the actual storage instance, which can be a database, a text file, or any other storage structure (default SQLite)

Near-RT RIC

E2 Terminator

RxReport

TxCommands

E2 node

E2 Node Terminator

Reporter

Report Trigger

Reporter

Report Trigger

Data Repository

Reporter

Report Trigger

Reports

Reports

Reports

Reports
The O-RAN ns-3 module
Architecture

Near-RT RIC

Data Repository

E2 node (eNB)
- E2 Node Terminator
  - Reporter
  - Report Trigger

E2 node (UE)
- E2 Node Terminator
  - Reporter
  - Report Trigger

Near-RT RIC
- E2 Terminator
  - RxReport
  - TxCommands
The O-RAN ns-3 module
Architecture

Near-RT RIC

Logical Modules (xApps)

- Default.m
- AdditionalLm_1
- AdditionalLm_n

• One default xApp available
• The RIC invokes the xApp(s) periodically

If Timer_xApp Expired
Execute all xApps

Near-RT RIC
E2Terminator

- RxReport
- TxCommands

E2 node (eNB)

- E2 Node Terminator
- Reporter
- Report Trigger

Data Repository

E2 node (UE)

- E2 Node Terminator
- Reporter
- Report Trigger

Reports

Reports
The O-RAN ns-3 module
Architecture

Near-RT RIC

Logical Modules (xApps)

- Default.m
- AdditionalLm_1
- AdditionalLm_n

• Once executed, each xApp queries the DR and performs its task instantaneously
• A processing time can be introduced per xApp

While Timer_{xAppProcessing} not expired
wait()
The O-RAN ns-3 module
Architecture

Near-RT RIC

Logical Modules (xApps)

- Default.m
- AdditionalLm_1
- AdditionalLm_n

Commands

Conflict Mitigation

Data Repository

E2 node (eNB)

- E2 Node Terminator
- Reporter
- Report Trigger

E2 node (UE)

- E2 Node Terminator
- Reporter
- Report Trigger

Once the command timer is expired or all commands are ready, the commands are sent to the CM which mitigates the conflicts.

If Timer\_Command expired or all commands ready sendReadyCommands()
The O-RAN ns-3 module
Architecture

Near-RT RIC

Logical Modules (xApps)
- Default.m
- AdditionalLm₁
- AdditionalLmₙ

Conflict Mitigation

The commands are then forwarded to the corresponding E2 nodes.

E2 node (eNB)
- E2 Node Terminator
- Reporter
- Report Trigger
- Data Repository
  - Query

E2 node (UE)
- E2 Node Terminator
- Reporter
- Report Trigger
- Data Repository
  - Query

Near-RT RIC E2Terminator
- RxReport
- TxCommands

Reports

Commands
The O-RAN ns-3 module
Architecture

Near-RT RIC

Logical Modules (xApps)
- Default.mML
- AdditionalLm_1ML
- AdditionalLm_nML

Conflicts Mitigation

Near-RT RIC
E2Terminator
- RxReport
- TxCommands

E2 node (eNB)
- E2 Node Terminator
- Reporter
- Report Trigger

Data Repository

E2 node (UE)
- E2 Node Terminator
- Reporter
- Report Trigger

An additional delay can be added at this point to represent the delay of the E2 interface.
The logging allows for storing logic log messages and outputs generated by the logic processing and conflict mitigation modules.
ns3-oran uses Open Neural Network Exchange (ONNX) for inferences

- ONNX defined to facilitate interoperability between various ML framework and tools.
- Both a file format and a set of operators
- Supported by PyTorch, MXNet but converters exist for TensorFlow and CoreML

Enabled by ONNX runtime (https://onnxruntime.ai/) linked through the ns-3 build process

- Allows access to ONNX models and query them directly via function calls as any other C++ Object => Fast compared to the usage of inter-process communication
Case study
Handovers: Scenario

- **2 Single-cell eNodeBs**
- **4 mobile UEs**
  - UE1 and UE4: restricted to the coverage of a single eNodeB
  - UE2 and UE3: moves into an area covered by both cells
- The goal is to **trigger efficient handover for UE2 and UE3**
- **Data Traffic**: Each UE: 290 kbit/s video stream download for 200 s

We compare the performances of **four approaches** using the following metrics:
- Packet Loss Rate for each UE
- Aggregated Throughput observed at the UEs
- Total Number of handovers

- eNb1 can serve a single UE without losing a single packet
- eNb2 can serve a single UE without losing a single packet

![Diagram showing the handover scenario with two single-cell eNodeBs and four mobile UEs.](image-url)
Case study
Handovers: the four approaches

- **Baseline**: No handover and UE in Configuration 1

- **Heuristic**: Use the Reference Signal Received Power (RSRP) of the UE’s serving cell and neighboring cells
  
  \[
  \text{If } \text{RSRP}_\text{neighborCell} > \text{RSRP}_\text{servingCell} \\
  \text{Handover to neighbor cell}
  \]

- **Distance**: Use the distance from the UE’s to the serving cell and neighboring cells
  
  \[
  \text{If } \text{distance}_\text{neighborCell} < \text{distance}_\text{servingCell} \\
  \text{Handover to neighbor cell}
  \]
  
  • The xApp collects the location of each UE

- **ML**: Deep learning Classification (4 configurations)
  
  - **Input**: distance of each UE to each eNodeB and the observed normalized packet loss of the UEs during the last 1 s
  
  - **Output**: Ranking of each configuration, which allows the selection of the one providing the lowest overall packet loss for the next 1 s
  
  - **Training**: One simulation for each configuration without any handover algorithms. Post-processing the data determines which configuration is best at any given time
  
  • The xApp collects the location of each UE as well as the packet loss and passes them to the classifier

---

**Table 1: Parameters of the Simulated Scenario**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated Time</td>
<td>210 s</td>
<td>s</td>
</tr>
<tr>
<td>Traffic</td>
<td>299 kbps</td>
<td></td>
</tr>
<tr>
<td>Pathloss Model</td>
<td>Cost231</td>
<td>NA</td>
</tr>
<tr>
<td>Max UE-to-eNodeB Distance</td>
<td>461 m</td>
<td></td>
</tr>
<tr>
<td>Scheduler</td>
<td>Round Robin</td>
<td>NA</td>
</tr>
<tr>
<td>O-RAN dependent parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LM Invocation Interval</td>
<td>1 s</td>
<td></td>
</tr>
<tr>
<td>UE/eNodeB Report Interval</td>
<td>1 s</td>
<td></td>
</tr>
<tr>
<td>ML training parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epochs</td>
<td>3</td>
<td>NA</td>
</tr>
<tr>
<td>Training Set Size</td>
<td>666 667</td>
<td>NA</td>
</tr>
<tr>
<td>Verification Set Size</td>
<td>333 333</td>
<td>NA</td>
</tr>
</tbody>
</table>
Case study
Results per approach

Approach Packet Loss

Approach Handover Count

Approach Aggregate Throughput

No delay added
Case study
Results per approach

- **Baseline:** Higher Loss (Avg Packet Loss 22.1%)
- The greatest improvements come from the O-RAN-based approaches, with the ML approach providing the greatest reduction in packet loss (Avg Packet Loss 3.5%)
- O-RAN-based approaches require additional handovers
Distance and ML approaches provide an overall higher throughput throughout the entire simulation.

Two O-Ran based approaches (Distance and ML) throughput greater than 1Mbit/s.

Baseline and heuristic dips below 610 kbit/s.

ML approach is able to provide increased performance but with fewer handovers than the distance approach.
Case study
Results for ML approach: additional delay

**Delay Packet Loss**

**Delay Handover Count**

**Delay Aggregate Throughput**

1s and 2s delay added
Case study
Results for ML approach: additional delay

- The larger the delay, the larger the packet loss
- The number of handovers remains unchanged
- Larger delays hinder throughput performance
All the components that contain logic modifiable by the end users have been modeled hierarchically, i.e., parent classes can take care of common actions and methods, and leave child models to focus on the logic itself.

**O-RAN with O-RAN Distance Approach**
All the components that contain logic that may be modified by end users have been modeled hierarchically, i.e., parent classes can take care of common actions and methods, and leave child models to focus on the logic itself.

To add O-RAN ML Approach

- Extend database API to store and retrieve application loss data
- Pass application loss report to Data Repository
- Store Log
- Conflict Mitigator
  - Single Command
  - Noop
- Logic Module
  - LTE 2 LTE Distance Based Handover
- Generates
- Logic Module No-op

Diagram:

- Data Repository
- E2 Terminator
- Reporter
- Near-RT RIC
- E2 Node
- Information Element
Conclusion and next steps

• ns3-oran makes it possible to quickly define, evaluate, and tune solutions and mechanisms for improving RAN performance based on O-RAN architecture

• ns3-oran is user-friendly and easily extendable

• Future work:
  • Implement O-RAN case studies
  • Use multiple xApps simultaneously
  • Extend ns3-oran to support NR Technology
  • Study the usage of ONNX to train models