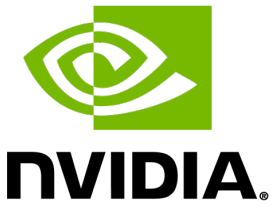


Aerial Testbed

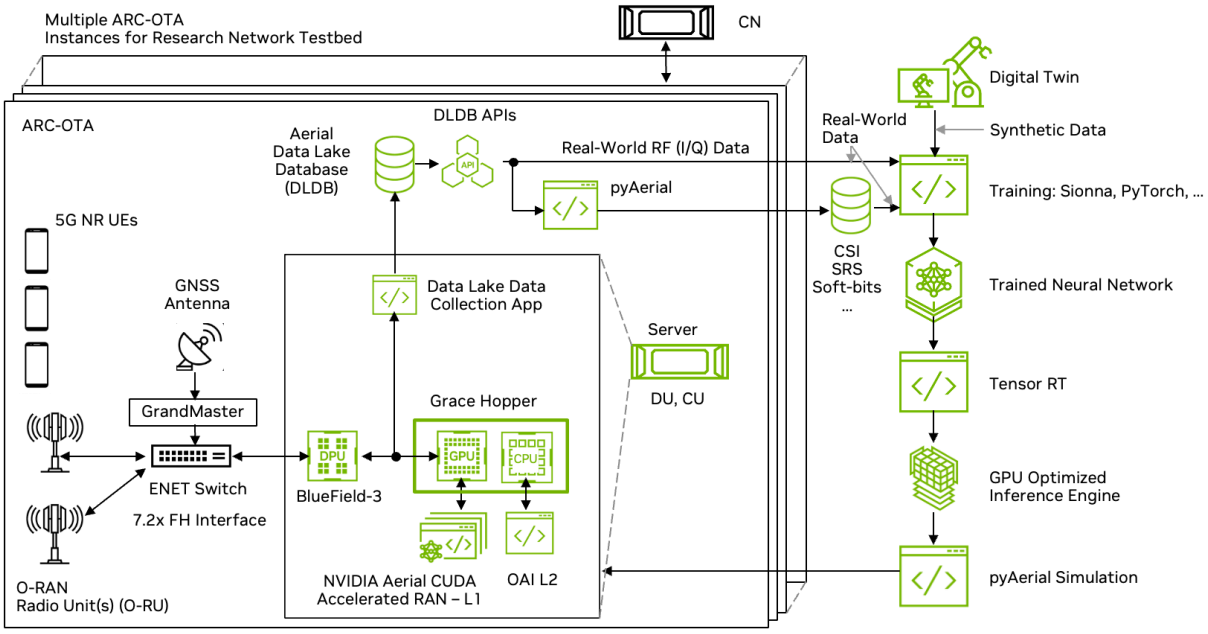
Release 1.0

NVIDIA Corporation

Apr 30, 2026

Contents

Chapter 1. Introduction



NVIDIA Aerial Testbed (ATB) is a highly capable and versatile platform for cellular research and commercial applications. It is a 3GPP- and O-RAN-compliant RAN and Core Network platform that interoperates with commercial UEs in cabled or OTA environments. Its software is comprised of Aerial CUDA-Accelerated RAN (ACAR) for O-DU-low and OpenAirInterface (OAI) for O-DU-high, CU, and the Core Network.

Key Features

- **Powerful Hardware**: Supports both Grace Hopper and DGX Spark, allowing developers to test and validate their solutions across different-sized platforms.
- **Open Source, Software-Defined**: ACAR and OAI are open source, allowing developers to understand and modify functionality to fit their needs.
- **GPU-Accelerated Stack**: Full inline acceleration of all PHY functions and select MAC functions provide high performance.
- **AI-native Architecture**: Allows NVIDIA's vast AI software offerings to be combined with RAN functionality, providing a powerful AI-RAN platform.
- **O-RAN Compliance**: Enables interoperation with different O-RUs and Test and Measurement equipment.

- **Versatile**: Can be operated fully OTA with commercial O-RUs and UEs, RF cabled in shielded environments or over eCPRI to O-RU/UE emulators. Supports soft-UEs for two-sided experimentation.
- **Data Collection**: Data Lake enables real-time collection of data from the RAN for offline analysis, training of AI models, or real-time usage by dApps.
- **dApps**: dApps (distributed Apps) extend the functionality of the RAN, allowing developers to add custom, real-time, and AI-powered functionality.

1.1. Software Release Manifest

The following table outlines the software versions used for ATB 1.0.

Component	Version
Aerial CUDA-Accelerated RAN	26-1
OAI (gNB, CN)	2026.w14
WNC 1220 O-RU (n78 and CBRS n48)	v1.9.0.

OAI artifacts can be found in the ATB1.0_integration branch.

Chapter 2. Release Notes

This page details new features, bug fixes, performance improvements, limitations, and documentation updates by release.

Note

With the Aerial Testbed 1.0 release, this project has been renamed from “ARC-OTA” to “Aerial Testbed”. The “ARC-OTA” name is still used to refer to legacy releases below.

2.1. Aerial Testbed Release 1.0 (April 2026)

2.1.1. New Features

- Aerial Testbed (ATB) is now supported on DGX Spark. An over-the-air testbed can now be implemented on DGX Spark. In this release single-cell operation is supported.
- dApp (distributed App) capabilities have been expanded. Users can now write a dApp in Python, C, CUDA or PyTorch and easily deploy on ATB. The user dApp can access various signals, for example front-haul time-frequency resource block I/Q samples, CSI, and other signals, in the PUSCH pipeline using the dApp subscription manager. The dApp framework enables real-time machine learning to support applications such as ISAC (Integrated Sensing and Communications) and spectrum management. Refer to the *Data Lake* and *dApp* sections of this documentation for full details.
- Data Lake has new capabilities to save the downlink transport block.
- The OAI GPU-accelerated Soft-UE is supported on Jetson-ORIN and DGX Spark with higher data rates and stability.
- ATB installation is now simplified with a new installation script. For step-by-step commands (for DGX Spark and Ubuntu Server), refer to the *Automated Installation of Aerial CUDA-Accelerated RAN* section in the installation guide.

2.1.2. Known Issues and Workarounds

- On occasion the L2 can stop scheduling, the UE will repeatedly perform RACH/random access and the following error codes 0x03 and 0x06 will be logged in the cubb logs. The errors are an indication that there is late L2 response. The gNB will need to be restarted.

- In a RF cabled or OTA 2 cell configuration, there are occasional failures in which UE traffic drops to zero in cell 2. The traffic in cell 1 is not impacted. The gNB will need to be restarted.
- In a 2-cell OTA configuration the DL and UL throughput is observed to be reduced in each cell when compared to a single cell configuration.
- When using 2-layers in the uplink with SRS enabled, high BLER can be observed. The gNB needs to be restarted.
- Notes on SSB block power:
  - If using Foxconn RU or WNC RU with digital-power-scaling legacy, the L1 `cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_FXN_GH.yaml` should be configured as follows:

    ```
    fs_offset_dl: 7
    fs_offset_ul: -5
    ```

    And the L2 `targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb-vnf.sa.band78.273prb.aerial.conf` should be configured as follows:

    ```
    ssPBCH_BlockPower = -34
    ```

  - If using digital-power-scaling o-ran, the L1 should be configured as follows:

    ```
    fs_offset_dl: 0
    fs_offset_ul: 0
    ```

    And the L2 should be configured as follows:

    ```
    ssPBCH_BlockPower = -11
    ```

- There is a bug in the WNC RU where, if multiple DUs send fronthaul (FH) packets, UEs will see MIB and SIB but will not receive Msg2; at the L1 you will only see CRC errors.

2.2. ARC-OTA Release 1.7 (January 2026)

2.2.1. New Features and Bug Fixes

- GPU (Jetson AGX) accelerated soft-UE: Support for 30 MHz RF BW. Limited data rates (30/30 Mbps DL/UL).
- Multi-UE support: DL and UL traffic to 100 UEs (using UE simulator over eCPRI)
- UL Heavy TDD Pattern (DSUUU): OTA validation with OAI L2+
- Data Lake:
  - Real-time gNB I/Q sample capture from 4 cells
  - Real-time soft UE data collection enables correlation across gNB and UE.
  - Data is accessible to E3 agent and in RAM database for use by dApps Framework.
  - The system can be configured to not capture data in the RAM database if the data is only to be consumed by the E3 agent.
  - MAC PDUs are now captured even when PUSCH CRC fails.

- Aerial CUDA-accelerated RAN software 25-3
- OAI gNB SW: ARC1.7_integration branch (2025.w46 baseline)

2.2.2. Performance Improvements

- TDD patterns: DDDDD DSUUU, DSUUU
- MIMO layers
  - DL: 4 layers
  - UL: 2 layers
- Single cell peak throughput
  - 1 UE:
    - DL: ~1.2 Gbps
    - UL: ~240 Mbps (DSUUU)
  - 4 UEs:
    - DL: ~1.1 Gbps
    - UL: ~135 Mbps

2.2.3. Deprecations

- Foxconn O-RUs are no longer supported.

2.2.4. Known Issues and Workarounds

- **Multiple UE support**: Aggregated throughput may be low when scheduling multiple UEs simultaneously. Throughput occasionally drops to zero after running traffic with multiple UEs for some time. The current workaround is to restart the test.
- SRS is only supported when PUSCH is simultaneously transmitted by the UE. SRS operation with 2L UL or operation with many UEs may have stability and performance issues after some time.

2.3. ARC-OTA Release 1.6 (August, 2025)

2.3.1. New Features and Bug Fixes

- 4T4R O-RUs
  - WNC 1220
  - WNC 3210
- Multi-UE support
  - 55 UEs attach and ping (UE simulator over eCPRI)
  - 2 UEs traffic

- Process SRS samples received from RU for use by L2 (4T4R; cabled)
- Aerial Data Lake: Real-time I/Q sample capture from 2 cells
  - This enables real-time capture of samples to DRAM for use in real-time processing.
    - Previously, real-time samples were captured to SSD and only enabled offline processing, such as for model training or visualization.
  - Real-time processed samples can now be used by algorithms to perform closed-loop actions on L1/L2 with low latency.
- Aerial CUDA-accelerated RAN software 25-2
- OAI gNB SW: ARC1.6_integration branch (2025.w29 baseline)

2.3.2. Performance Improvements

- TDD pattern: DDDDD DSUUU
- MIMO layers
  - DL: 4 layers
  - UL: 1 layer
- Single cell peak throughput
  - 1 UE:
    - DL: ~1.2 Gbps
    - UL: ~126 Mbps
  - 2 UEs:
    - DL: ~1.4 Gbps
    - UL: ~75 Mbps
- 2 cell aggregated peak throughput (1 UE per cell)
  - DL: ~1.4 Gbps
  - UL: ~180 Mbps

2.3.3. Deprecations

- Dell PowerEdge R750 Server + A100X HW is no longer supported.
- Gigabyte Edge E251-U70 Server is no longer supported.

2.3.4. Known Issues and Workarounds

- There are connection drops after running iPerf traffic for some time. The UE will re-initiate the connection autonomously.
- Data stalls with traffic to multiple UEs: Throughput occasionally drops to zero after running traffic with multiple UEs for some time. The current workaround is to restart the test.

2.4. ARC-OTA Release 1.5 (July, 2024)

2.4.1. New Features and Bug Fixes

- Grace Hopper integration
  - OAI ARM CPU support
  - Hopper GPU cuBB support
- CBRS O-RU
- CBRS RU Interop
- Multi-UE support
- Multi-UE CSI dataset blueprint (using Aerial Data Lake and PyAerial)
- Updated OAI branch 2024.w21+ARC1.5
- Multi-L2 Support (2 Cells)

Tip

For additional context, you can also review the artifacts in the `2024.w21+ARC1.5` branch.

Note

The Grace Hopper platform requires OAI CN version `2024-June`.

2.4.2. Performance Improvements

- MIMO layers
  - DL: 2 layers -> 4 layers
- Peak throughput
  - SMC-GH
    - DL: ~460Mbps -> ~1.03Gbps
    - UL: ~112Mbps -> ~125Mbps
  - Dell R750 / Gigabyte Edge E251-U70
- Multi-L2 (2 Cell) throughput
  - SMC-GH
    - DL: ~700Mbps (per Cell)
    - UL: ~105Mbps (per Cell)

2.5. ARC-OTA Release 1.3 (May, 2024)

2.5.1. New Features and Bug Fixes

- Full support for master gitlab repo gitlab.eurecom.fr/oai/ making it available for use, modification and distribution. Refer to the *Licensing* page for the OAI license.
  - **OAI_Aerial private branch is deprecated and will no longer be maintained.**
- Developer extension - Sterling k8 service management and monitoring (refer to this document for more details)
- Updated OAI branch `2024.w15` with a standalone patch (`w15.4`).

2.5.2. Performance Improvements

- Frame structure and slot format
  - DDDSU -> DDDSU + DDDDDDSUUU
- Bi-directional UDP Traffic
  - > 3.5 hours exercised -> > 4.0 hours exercised

2.6. ARC-OTA Release 1.2 (March, 2024)

2.6.1. New Features and Bug Fixes

- Support for the Dell R750 platform to host gNB
- Support for NVIDIA Converged Accelerators A100X
- Continued support for Gigabyte Edge E251-U70 with NVIDIA discrete cards A100 GPU, CX6-DX SmartNIC
- OAI_Aerial_v2.2.1 updated to OAI_Aerial_v2.2.2

2.7. ARC-OTA Release A1.1 (January, 2024)

2.7.1. New Features and Bug Fixes

- Kernel cmdline configuration updated.
- Updates to the core assignment in the Aerial configuration.
- Updates to PTP and phc2sys core assignment.
- Changes to phc2sys cmdline.
- Changes to the L2 Docker run cmd to use all non-isolated cores.
- Changes to the L2 configuration, max DL MCS defaults to 25.

- Removal of unnecessary ORU firmware installation step because of Foxconn firmware default updates.
- OAI_Aerial_v2.0 updated to OAI_Aerial_v2.2.1 throughout.

2.7.2. Known Issues and Workarounds

- 256 QAM is not supported on ARC-OTA. You must disable 256 QAM support by issuing the following command at the gNB command license:

```
--gNBs.[0].force_256qam_off
```

Chapter 3. Product Description

Aerial Testbed is an AI/ML enabled end-to-end, real-time Over-The-Air (OTA) testbed for AI-RAN research. As shown in the figure below, ATB is an on-prem edge-cloud datacenter that is built on the NVIDIA Aerial CUDA-Accelerated RAN in-line accelerated L1, integrated with the OpenAirInterface (OAI) Software Alliance L2, and Core Network (CN). The Aerial CUDA-Accelerated RAN L1 runs on the Hopper GPU and the OAI L2 runs on the Grace CPU. The default L2 configuration assumes the CN is run on the same host as the L1 and L2, but the L2 also supports running the CN on a separate x86 or ARM server. An NVIDIA NIC connects via a fronthaul switch to one or more O-RUs using O-RAN 7.2x for single or multi-cell operation.

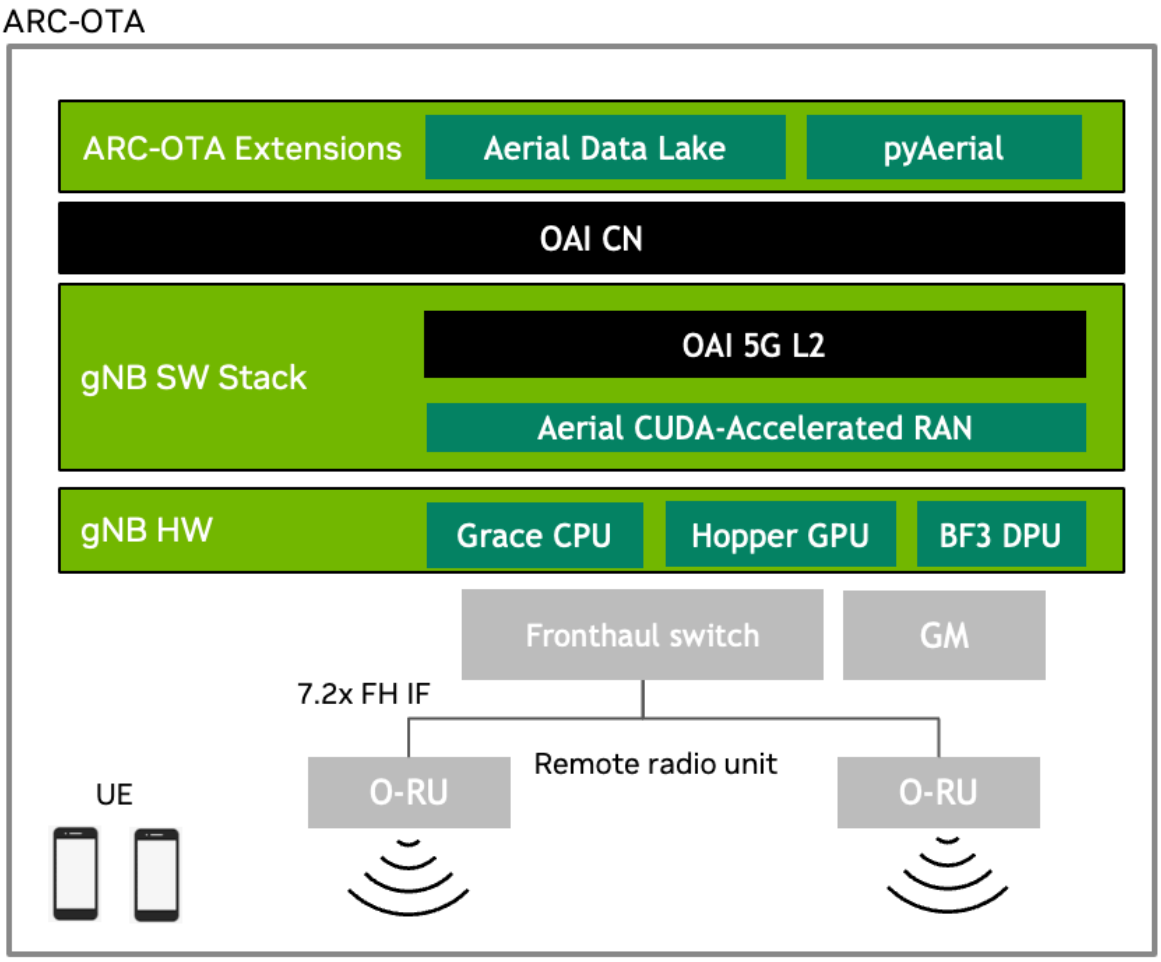
NVIDIA Aerial CUDA-Accelerated RAN L1 and OAI L2 stacks are open source. Researchers can bring their innovations to reality through customization of the modulation, coding and signal processing algorithms in the data and control channels of the air interface. With source for L2 machine learning (ML) algorithms, deep reinforcement learning (DRL) can be implemented in the MAC and scheduler.

With an ATB testbed you can test ML algorithms at all layers of the stack. You can bring ML to the physical layer, to layer 2, and benchmark them in a live network. As real O-RUs are used in the testbed, your algorithms can be verified and benchmarked in the context of real-world wireless channels, in addition to all the non-idealities present in a physical gNB such as power amplifier non-linearities, RF gain and phase mismatch, and other imperfections in the analog electronics. ATB can be used in conjunction with real-time channel emulators and UE emulators to test algorithms with traditional 3GPP stochastic channel models in addition to using site-specific models generated by RF ray tracing in a digital twin, such as NVIDIA Aerial Omniverse Digital Twin.

ATB is built with an eye to enabling AI/ML research. It supports the capture of OTA data for use in training pipelines. Data collection is facilitated using NVIDIA Aerial Data Lake, which collects uplink I/Q samples from O-RU(s) over the 7.2x fronthaul interface and writes them to a database. FAPI meta information exchanged between L1 and L2 is also collected and populated in the database and can be used for indexing into and extracting data from the Data Lake database.

While the uplink I/Q samples and L2 meta information are useful for some types of algorithm development, each type of ML, or for that matter non-ML, algorithm design requires a data set tailored to the use case at hand. This is where NVIDIA pyAerial helps. While there are multiple uses for pyAerial, one application is for generating data sets corresponding to any node in the cuPHY PUSCH pipeline. pyAerial brings cuPHY CUDA kernels to Python. It is a library of cuPHY L1 kernels that have been provisioned with Python APIs. It is straightforward for a researcher to, for example, assemble a complete PUSCH pipeline using pyAerial blocks. Since under the pyAerial API the blocks are invoking the same CUDA code that is employed in the real-time cuPHY L1, the pyAerial pipeline is bit-equivalent to the pure CUDA cuPHY pipeline. You might want access to, for example, the input and output samples of cuPHY minimum mean square error (MMSE) channel estimator. You can simply instrument your pyAerial Python code with file I/O operations for each node of interest in the pyAerial graph.

The figure below shows the E2E architecture and software stack for Aerial Testbed.



3.1. Key Features and Specifications

The configuration and capabilities of ATB 1.0 are outlined in the following sections.

Feature	Value
Number Antennas	4T4R
Number of Component Carriers	1x 100MHz carrier
Subcarrier Spacing (PDxCH; PUxCH, SSB)	30 kHz
FFT Size	4096
MIMO layers	DL: 4 layers; UL: 1 layer
Duplex Mode	Release 15 SA TDD
Number of RRC connected UEs	Up to 100
Number of UEs/TTI	16
Frame structure and slot format	DDDDDDSUUU
	DSUUU
User plane latency (RRC connected)	< 10ms one way for DL and UL mode
Synchronization and Timing	IEEE 1588v2 PTP; SyncE; LLS-C3
Frequency Band	n78, n48 (CBRS)
Max Transmit Power	22 dBm at RF connector
Peak throughput	Refer to the *Release Notes* for the latest peak throughput values.
Bi-directional UDP Traffic	> 10+ hours exercised (SMC-GH)

Tip

To learn how KPIs have changed from last release, refer to the *Release Notes*.

3.1.1. 5G NR gNB Features

Component	Capabilities
gNB PHY	Refer to the Aerial CUDA-Accelerated RAN cuBB documentation
gNB MAC	Refer to the Aerial CUDA-Accelerated RAN cuBB documentation

3.1.2. 5G Core Features

AMF	Features	NGAP AMF status indication (3GPP TS 38.413)
		Add UE Retention Information support (3GPP TS 38.413)
		Support of Location services with LMF and AMF (3GPP TS 29.518, 3GPP TS 38.413, 3GPP TS 23.502)
		Update NAS with Rel 16.14.0 IEs: Refactor code for Encode/Decode functions; cleanup NAS library (3GPP TS 24.501)
	Fixes	Fix typo for N1N2MessageSubscribe (3GPP TS 29.518)
		Fix issue when receiving PDU session reject from SMF (3GPP TS 29.518, 3GPP TS 23.502)
	Technical Debt	Reformatting of the SCTP code
		Refactor promise handling
		Removing dependencies to libconfig++ (Only YAML file can be read as configuration)
SMF	Features	Add N1/N2 info in the message response to AMF if available (3GPP TS 29.502)
	Fixes	Add connection handling mechanism between NRF and SMF
	Technical Debt	Refactor SMF PFCP associations to use UPF profile
UDM	Fixes	Add connection handling mechanism between NRF and UDM
UDR	Technical Debt	Fixed builds
		Add connection handling mechanism between NRF and UDR
		Improve MongoDB support
Common		New HTTP Client library (CPR) for all the NFs
		Support mobility registration update procedure (3GPP TS 23.502)

3.1.3. 5G Fronthaul Features

RU Category	Category A
FH Split Compliance	7.2x with DL low-PHY to include Precoding, Digital BF, iFFT+CP and UL low-PHY to include FFT-CP, Digital BF
FH Ethernet Link	25Gbps x 1 lane
Transport encapsulation	Ethernet
Transport header	eCPRI
C Plane	Conformant to O-RAN-WG4.CUS.0-v02.00 7.2x split
U Plane	Conformant to O-RAN-WG4.CUS.0-v02.00 7.2x split
S Plane	Conformant to O-RAN-WG4.CUS.0-v02.00 7.2x split
M Plane	Conformant to O-RAN-WG4.CUS.0-v02.00 7.2x split
RU Beamforming Type	Code book based

3.2. Data Collection on Aerial Testbed with Data Lake

Aerial CUDA-Accelerated RAN supports data capture for offline processing using logging, which is useful for debugging and performance analysis. It also captures richer control and data plane data in real time in Data Lake for offline and real-time usage to enhance debugging and performance analysis, to train AI/ML models, and to be used by dApps, written by 3rd parties to achieve new RAN functionality.

Data Lake is a real-time component of ACAR running on the O-DU that captures L1 and L2 data from O-RUs and the O-DU.

It consists of three parts (refer to figure below):

1. The Data Collection App (DCA) running on the CPU
2. The Data Lake Database (DLDB)
3. DLDB APIs used for retrieving data from the DLDB

3.2.1. Key Features

Data Lake has the following features:

Real-time capture of Fronthaul I/Q samples

The data passed to L2 via RX_Data.Indication and UL_TTI.Request are exported to the database.

API access to the database

Scalable and time coherent over arbitrary number of BSs

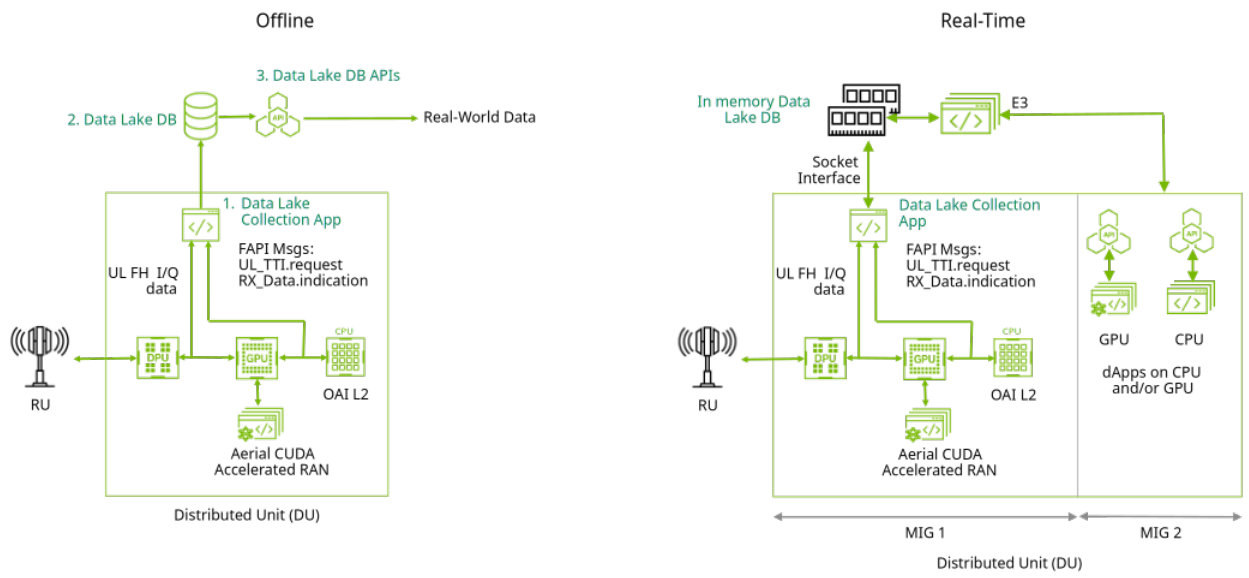


Fig. 1: Data Lake components for offline and real-time data access

The data collection app runs on the same CPU that supports the DU. It runs on a single core, and the database runs on free cores. Because each gNB is responsible for collecting its own uplink data, the collection process scales as more gNBs are added to the network testbed. Database entries are time-stamped so data collected over multiple gNBs can be used in a training flow in a time-coherent manner.

Use in conjunction with `pyAerial`_ to generate training data for neural network physical layer designs

Data Lake can be used in conjunction with pyAerial. Using the Data Lake database APIs, pyAerial can access RF samples in a Data Lake database and transform those samples into training data for all the signal processing functions in an uplink or downlink pipeline.

Use by dApps

dApps can use real-time data from the database to analyze system performance and/or to trigger real-time actions on the gNB based on the analysis.

3.2.2. Example: Training Data Generation Using pyAerial

Uplink I/Q data from one or more O-RAN Radio Units (O-RUs) is delivered to GPU memory where it is both processed by the L1 PUSCH baseband pipeline and delivered to host CPU memory. The Data Lake collector process writes the I/Q samples to the Data Lake database in the fh table.

The fh table has columns for SFN, Slot, IQ samples as fhData, and the start time of that SFN.slot as TsTaiNs.

The collector app saves data that the L2 sent to L1 to describe UL OTA transmissions in UL_TTI.Request messages as well as data returned to the L2 via RX_Data.Indication and CRC.Indication. This data is then written to the fapi database table. These messages and the fields within them are described in SCF 5G FAPI PHY Spec version 10.02, sections 3.4.3, 3.4.7, and 3.4.8.

Each gNB in a network testbed collects data from all O-RUs associated with it. That is, data collection over the span of a network is performed in a distributed manner, each gNB is building its own local

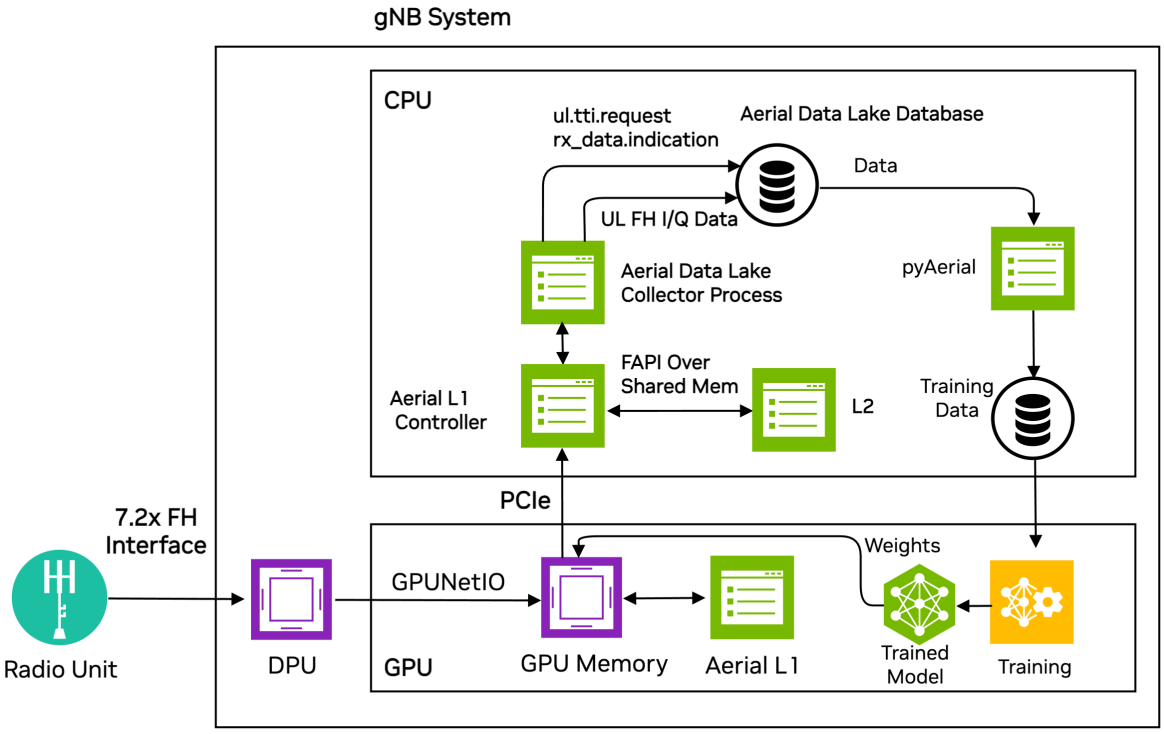


Fig. 2: Data Lake data capture and usage by pyAerial for model training

database. Training can be performed locally at each gNB, and site-specific optimizations can be realized with this approach. Since the data in a database is time-stamped, the local databases can be consolidated at a centralized compute resource and training performed using the time aligned aggregated data.

In cases where the PUSCH pipeline was unable to decode due to channel conditions, retransmissions can be used as ground truth as long as one of the retransmissions succeeds, allowing the user to test algorithms with better performance than the originals.

The Data Lake database storage requirements depend on the number of O-RUs, the antenna configuration of the O-RU, the carrier bandwidth, the TDD pattern and the number of samples to be collected. Collecting IQ samples of 1 million transmissions from a single RU 4T4R O-RU employing a single 100MHz carrier will consume approximately 660 GB of storage.

The Data Lake database comprises the fronthaul RF data. However, for many training applications access to data at other nodes in the receive pipeline is required. A pyAerial pipeline, together with the Data Lake database APIs, can access samples from an Data Lake database and transform that data into training data for any function in the pipeline.

The figure below illustrates data ingress from a Data Lake database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft de-mapper.

Information about installing and using Data Lake and pyAerial can be found in the Aerial CUDA-Accelerated RAN documentation.

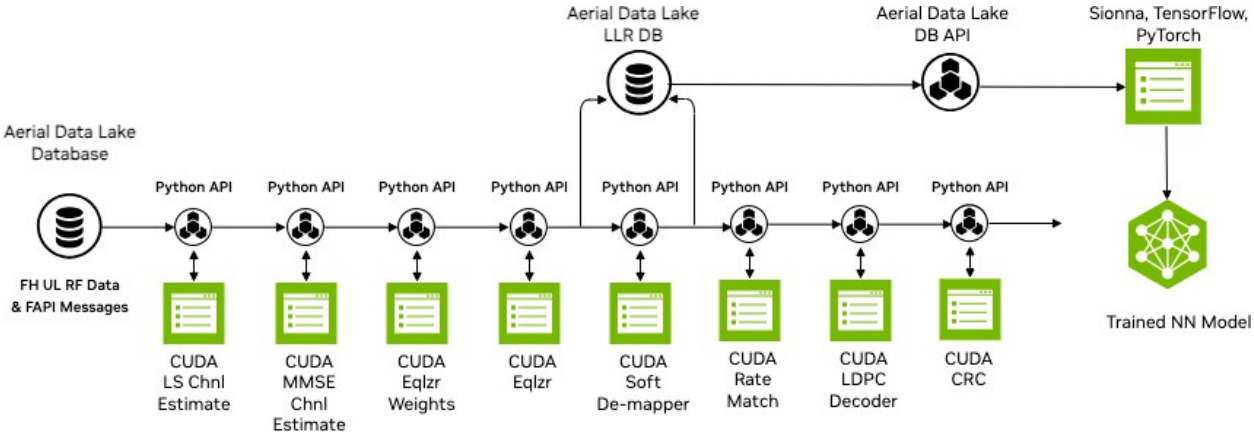


Fig. 3: pyAerial and Data Lake data flow for building training datasets for a neural network soft de-mapper

3.3. Extending RAN functionality with dApps

dApps are real-time, 3rd party applications that can analyze L1/L2 data from the RAN and use it to control RAN functionality. ACAR captures data using Data Lake and implements a dApp Framework through which dApps can register/deregister themselves, subscribe to and access Data Lake data, and send commands to control ACAR behavior (refer to figure below). dApps communicate with the dApp Framework using a pre-standard version of the E3 interface and E3 Application Protocol (E3AP).

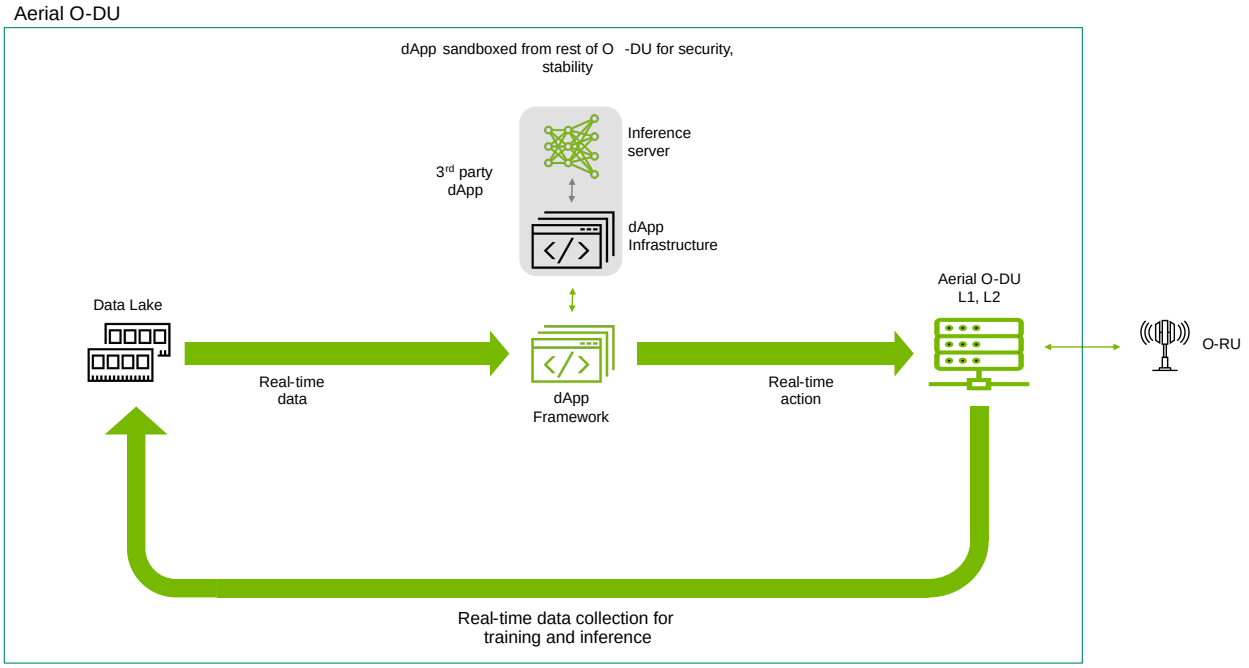


Fig. 4: dApp: High level operation

dApps run on the O-DU and extend its functionality. On GPUs that support MIG (Multi-Instance GPU), dApps and ACAR run in different MIG instances, providing process and memory isolation while maintaining low latency data exchange via CPU shared memory. Running on the same GPU-based HW as

ACAR enables dApps to run low latency AI/ML inference to analyze data from Data Lake and to determine actions to be sent to control the RAN.

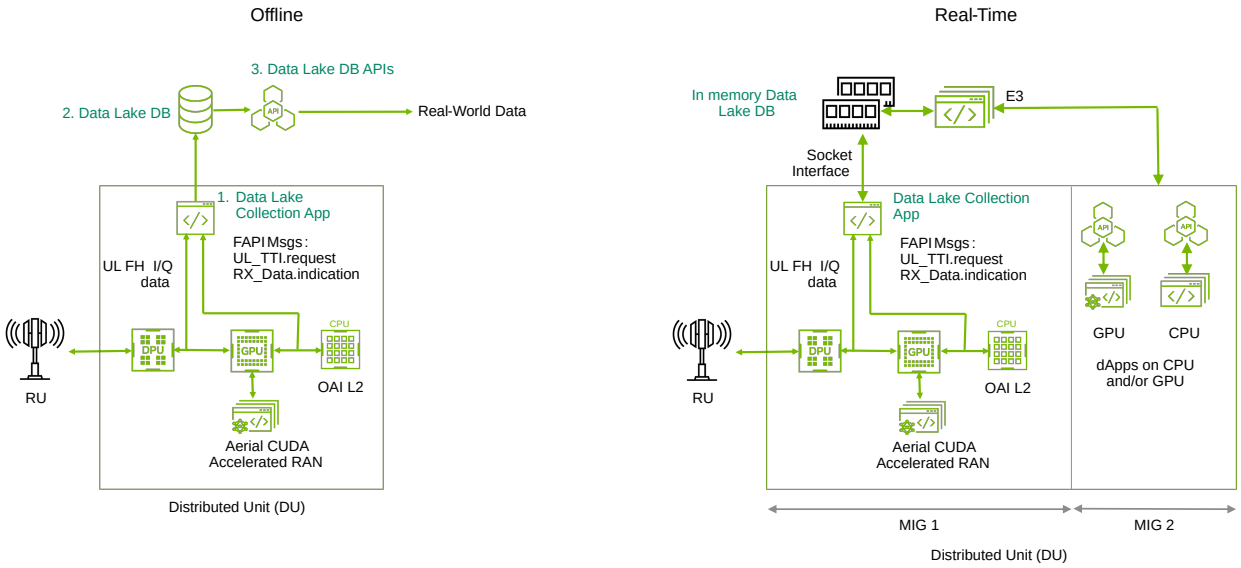


Fig. 5: dApp hardware and software architecture

3.3.1. Software components

dApp functionality is split among 4 main components in the O-DU, as shown in figure below.

1. DU-Low, which runs L1 pipelines, Data Lake and an E3 Agent
2. DU-High, which runs L2 and higher layer components. OpenAirInterface is used as an example L2+ stack
3. CPU Shared Memory (CPU SHM)
4. One or more co-located dApps

3.3.1.1 DU-Low

DU-Low functionality is provided by Aerail CUDA-Accelerated RAN (ACAR). The figure above shows three main components:

1. **CUDA Pipeline**: The figure shows a single PUSCH L1 pipeline for simplicity, which receives FH (Fronthaul) data from an RU and processes it into CRC-checked data blocks
2. **Data Lake**: Captures FH I/Q samples and other L1/L2 data from intermediate processing blocks and metadata. The data is captured in real time to memory. Data writes alternate between 2 buffers (ping and pong). When one buffer is full, writes begin to the other buffer and data from the full buffer is read out to a ClickHouse database configured in RAM tables or in SSD storage.
3. **E3 Agent**: Interfaces ACAR functionality with one or more dApps. It runs alongside the real-time data path in DU-Low, receives notifications when new uplink data is available, and exposes that data to subscribed dApps through the E3 interface. For small data items, it can send values directly in E3 indication messages; for larger data objects such as I/Q samples or channel estimates, it sends metadata and shared-memory references so the dApp can read the data efficiently. It also handles E3AP procedures such as setup, subscription, indication, and control,

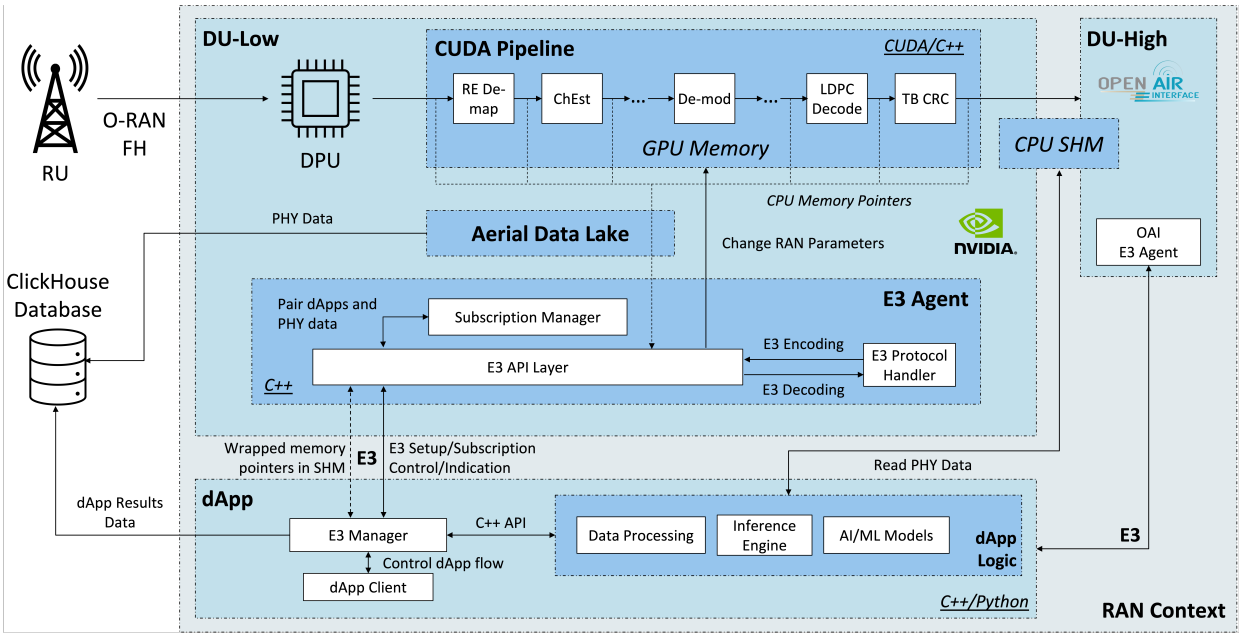


Fig. 6: dApp components

enabling dApps to discover data, receive real-time updates, and optionally send control commands back to the RAN.

3.3.1.2 CPU SHM (Shared Memory)

Shared memory is used to expose real-time RAN data to dApps with low latency and minimal overhead. Selected uplink data is copied from GPU memory into pinned host memory and organized in ping-pong buffers. Instead of sending large payloads through control messages, the framework sends dApps references to the relevant shared-memory location, including the data type, buffer, and offset. This lets dApps read large structures such as I/Q samples and channel estimates efficiently while keeping the critical PHY processing path decoupled from dApp execution.

3.3.1.3 DU-High

DU-High can interact with dApps through the E3 interface for data consumption and control actions similar to how they interact with DU-Low as long as it supports E3.

3.3.1.4 dApp sub-components

A dApp itself consists of 3 main sub-components (refer to the figure below):

- **E3 Manager**: The core engine that handles E3AP protocol, multi-agent communication, shared memory, and application dispatch.
- **dApp Client**: An external control interface for managing the dApp lifecycle: listing agents, subscribing to telemetry, unsubscribing, and querying status.
- **dApp Logic**: The application-specific processing logic (e.g., PRB power calculation) including inference engine. Each app has its own config, Dockerfile, models, and build system. Different inference engines are supported, such as Triton Inference Server, Python and TensorRT, and multiple models can be used, hosted in a model repository.

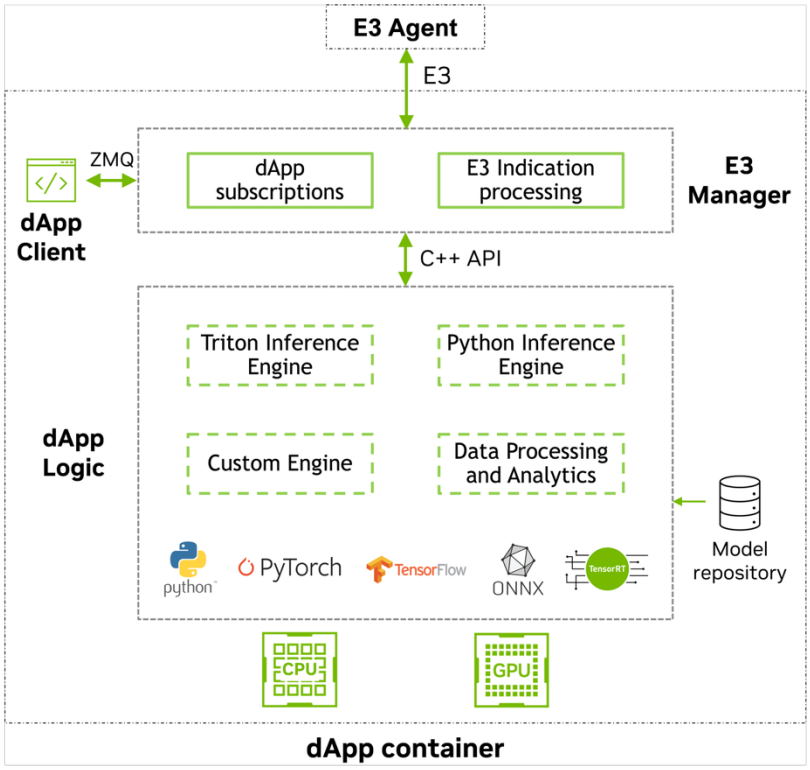


Fig. 7: dApp reference implementation

3.3.1.5 E3 Agent-Manager relationships

The figure below shows how multiple Agents can connect to multiple Managers.

It shows how a single E3 Agent can connect to multiple dApps (i.e., E3 Managers), each of which may subscribe to different data from the Agent and control different aspects of its behavior, likely based on different dApp logic.

Similarly, a single dApp (i.e., E3 Manager) may connect to different E3 Agents, accessing different data and controlling different behaviors, potentially based on different dApp logic.

3.3.2. Available Data Streams

The E3 Agent exposes the following uplink data streams through the NVIDIA KPM Service Model. dApps can subscribe to any combination of these to receive real-time RAN data for inference, analytics, or custom processing.

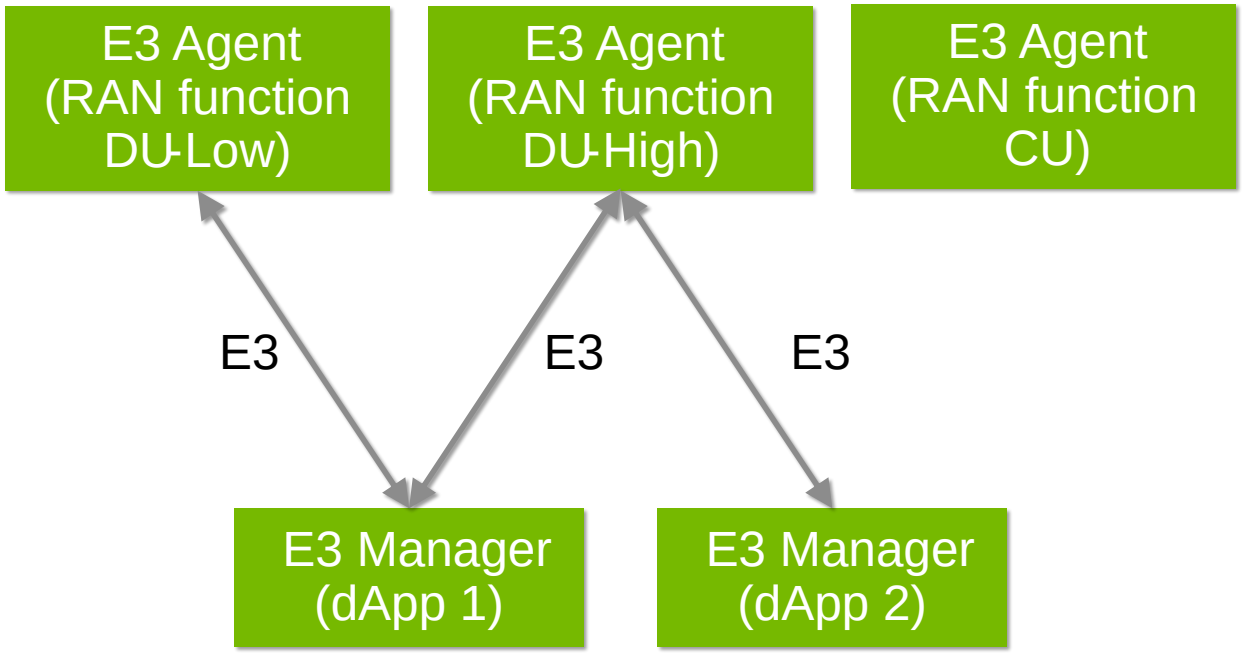


Fig. 8: Multiple Agent-Manager connections

Category	Data	Description
IQ & Channel	IQ samples, DMRS	Raw fronthaul IQ, DMRS channel estimates, PUSCH estimates, decoded PUSCH data (shared memory)
Frame info	SFN, Slot, Timestamp	System Frame Number, slot index, agent-side nanosecond timestamp
Cell & Antenna	Cell ID, RX antennas, BS antennas, Cells	Physical cell ID, antenna counts, Cells
Channel quality	RSRP, RSSI, CQI, MCS index, QAM order	PHY-layer measurements and modulation parameters
PUSCH	TB CRC fail, CB errors, CB count	Transport/code block error indicators
Resource allocation	RB start, RB size, Symbols, Subcarriers, MIMO layers	Frequency/time resource assignment

This list reflects the currently supported telemetry streams. Additional data streams and service models may be added in future releases. Refer to the NVIDIA Sample Apps repository for the full per-field telemetry ID table and shared memory layout details.

Future releases of E3 Agents may also expose other Service Models with different RAN Function IDs, providing additional telemetry IDs and/or control IDs.

3.3.3. References

- NVIDIA Sample Apps repository, containing sample dApps to show how to build, install, configure, test and run low-latency, inference-capable dApps to collect and consume data in real-time from ACAR. https://github.com/NVIDIA/aerial-sample-apps/tree/main/dapps
- [1] D. Villa, M. Belgiovine, N. Hedberg, M. Polese, C. Dick, and T. Melodia, "Programmable and GPU-Accelerated Edge Inference for Real-Time ISAC on NVIDIA Aerial Testbed," arXiv:2512.06493 [cs.NI], 2026. [pdf](https://arxiv.org/pdf/2512.06493)
- [2] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dApps: Distributed Applications for Real-time Inference and Control in O-RAN," *IEEE Communications Magazine*, 2022. [pdf](https://arxiv.org/pdf/2203.02370.pdf)
- [3] Northeastern University, NVIDIA, and Mavenir, "dApps Architecture and Interfaces," *O-RAN next Generation Research Group (nGRG)*, Research Report, 2025, report ID: RR-2025-05, v2.0. [pdf](https://mediastorage.o-ran.org/ngrg-rr/nGRG-RR-2025-05-dApps%20Architecture%20and%20Interfaces-v2.0.pdf)
- [4] Northeastern University, NVIDIA, Mavenir, MITRE, and Qualcomm, "dApps for Real-Time RAN Control: Use Cases and Requirement," *O-RAN next Generation Research Group (nGRG)*, Research Report, Oct 2024, report ID: RR-2024-10. [pdf](https://mediastorage.o-ran.org/ngrg-rr/nGRG-RR-2024-10-dApp%20use%20cases%20and%20requirements.pdf)
- [5] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese, and T. Melodia, "dApps: Enabling Real-Time AI-Based Open RAN Control," *Computer Networks*, vol.269, pp. 111342, 2025. [pdf](https://www.sciencedirect.com/science/article/pii/S1389128625003093)

3.4. Product Blueprints

To ease developer onboarding, this section provides reference blueprints with key ingredients for creating a full-stack tested product prototype.

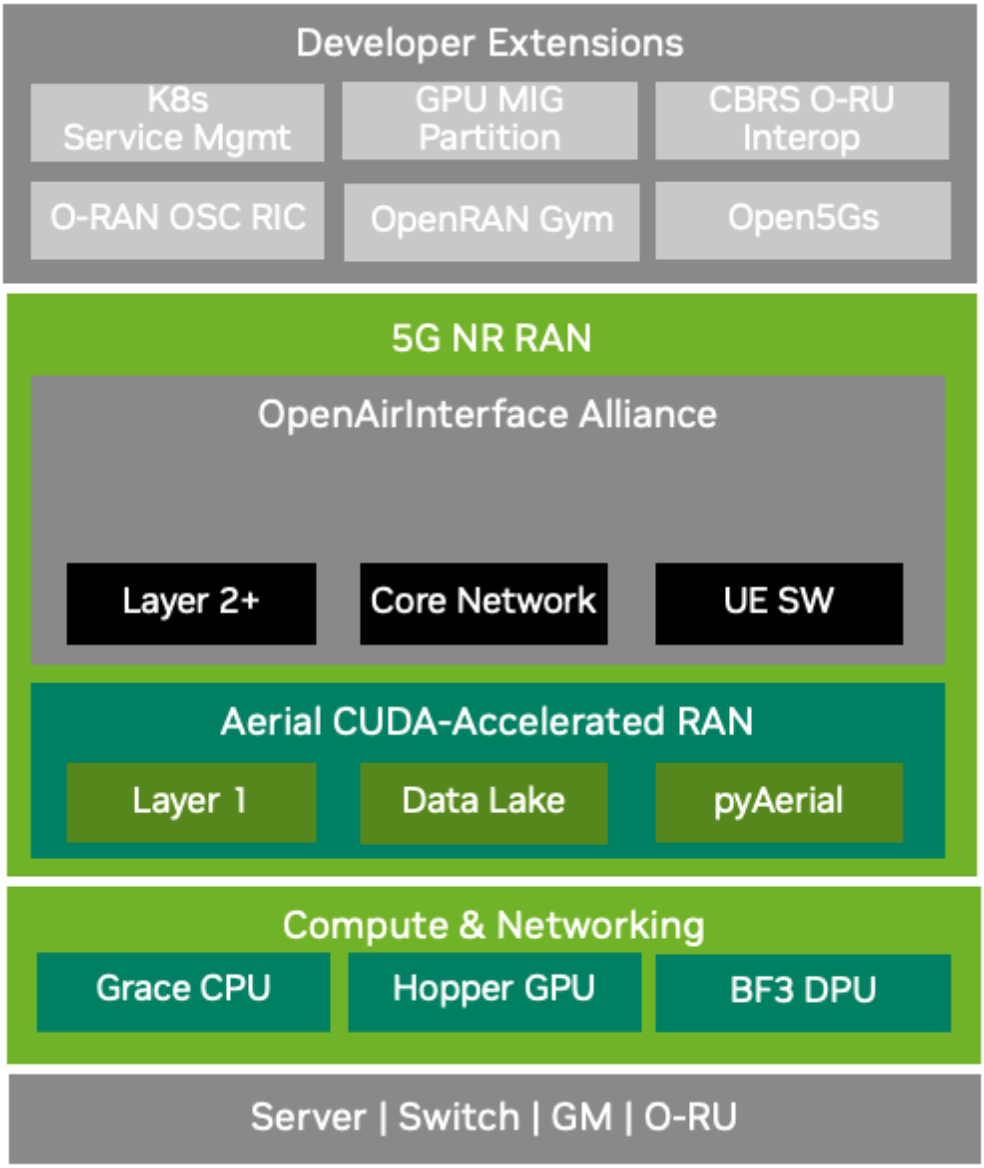
ATB with AI-RAN integration offers several advantages:

- **Real-world validation**: Many assumptions made for analytical and simulation studies can be rigorously tested in a real OTA network environment, now including AI-enhanced RAN functionality.
- **Comprehensive experience sharing**: Our extensive experience in innovation labs, including design, setup, deployment, and integration of tools and frameworks, is made available to all developers, incorporating insights from AI-RAN implementations
- **Qualified components**: Hardware components and software configurations have undergone rigorous qualification processes, addressing potential difficulties and pitfalls, now including AI-RAN accelerated computing platform
- **Controlled experimentation**: Developer variations in lab experimental networks are limited to environment variability, transmission power, attenuation, and a select set of variables
- **Full-stack programmability**: ATB provides complete access to source code, allowing developers to onboard any experiments with quick-turnaround validation and benchmarking results, now extended to AI-RAN applications
- **Cutting-edge technology**: Built on principles of disaggregation, virtualization, software-defined systems, adaptability, and O-RAN specifications, ATB serves as a true advanced wireless developer launchpad

- **AI-RAN integration**: The platform now supports concurrent AI and RAN processing, enabling developers to explore multi-tenancy and orchestration capabilities, maximizing capacity utilization
- **Energy efficiency**: AI-RAN integration allows for optimized energy consumption without compromising performance, enabling developers to test and validate energy-efficient network designs
- **Enhanced performance**: Developers can leverage AI-enhanced functionalities such as improved throughput, handover speed, and network anomaly detection

As we look to leverage, extend, and innovate, the key guiding attributes for these blueprints focus on creating a reliable, stable, performant, and scalable experimental radio network that empowers developers to push the boundaries of wireless technology.

3.4.1. Full-Stack Innovation



Components	Feature
COTS hardware	COTS infrastructure composed of accelerated compute, virtualization, radios, fronthaul networking, and precision timing.
Virtualization	vRAN workloads from NVIDIA and OAI
AI/ML Frameworks	Data Lake + pyAerial for AI/ML frameworks: RF / IQ data + FAPI
Standards	3GPP Release 15 + O-RAN 7.2 split P5G on-prem lab network
Developer Extensions	For more information on developer contributions, refer to the *Developer Zone* section

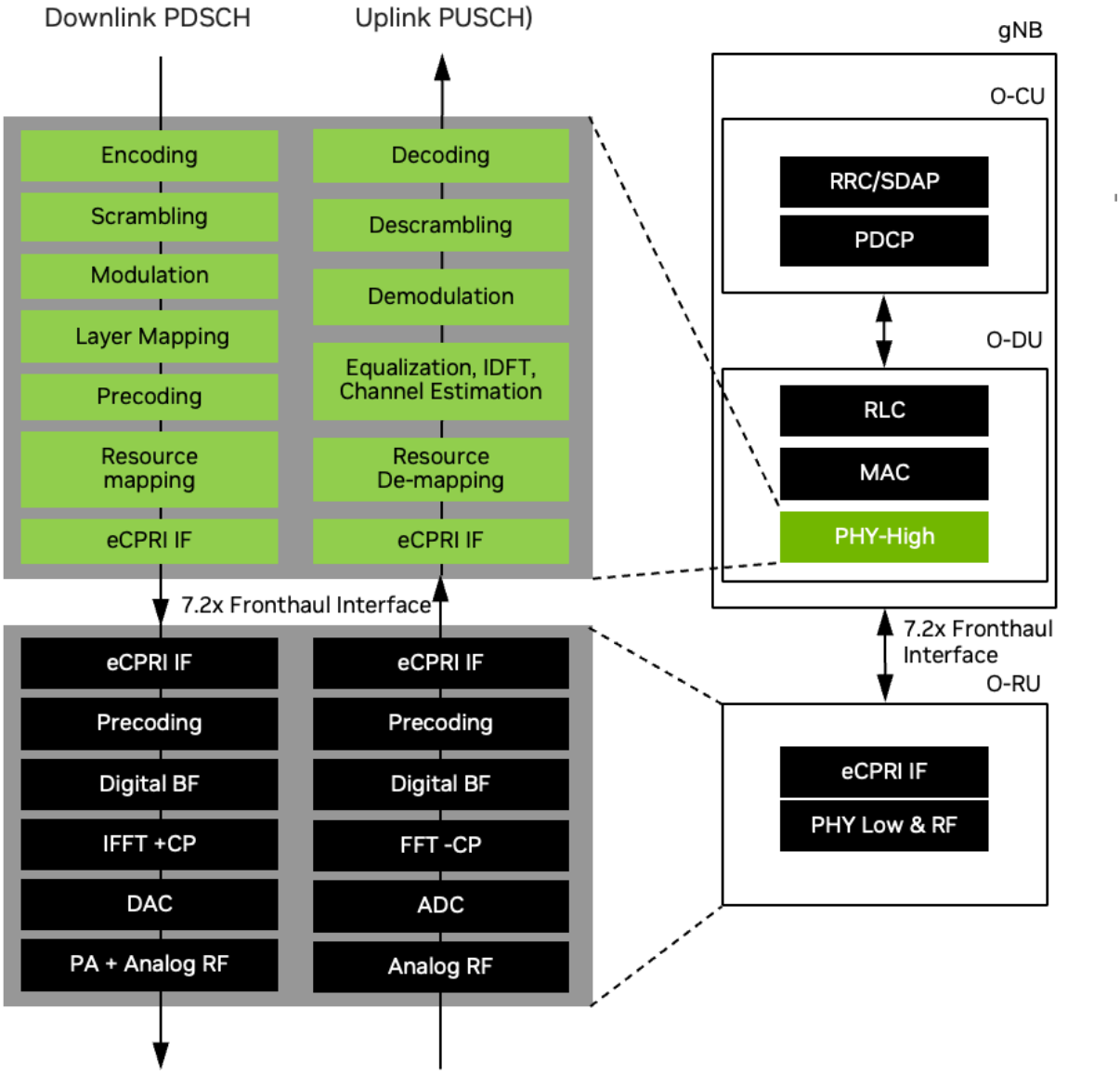
Note

We welcome developer contributions through extensions and plugins–for community benefits and to accelerate pace of innovations!

3.4.2. Aerial Testbed and O-RAN

The O-RAN framework is designed to foster openness, programmability, automation, intelligence, and the decoupling of hardware and software through standardized, interoperable interfaces. It champions a multi-vendor and multi-stakeholder ecosystem within a cloud-native and virtualized wireless infrastructure, enhancing the efficiency of RAN deployment, operation, and maintenance.

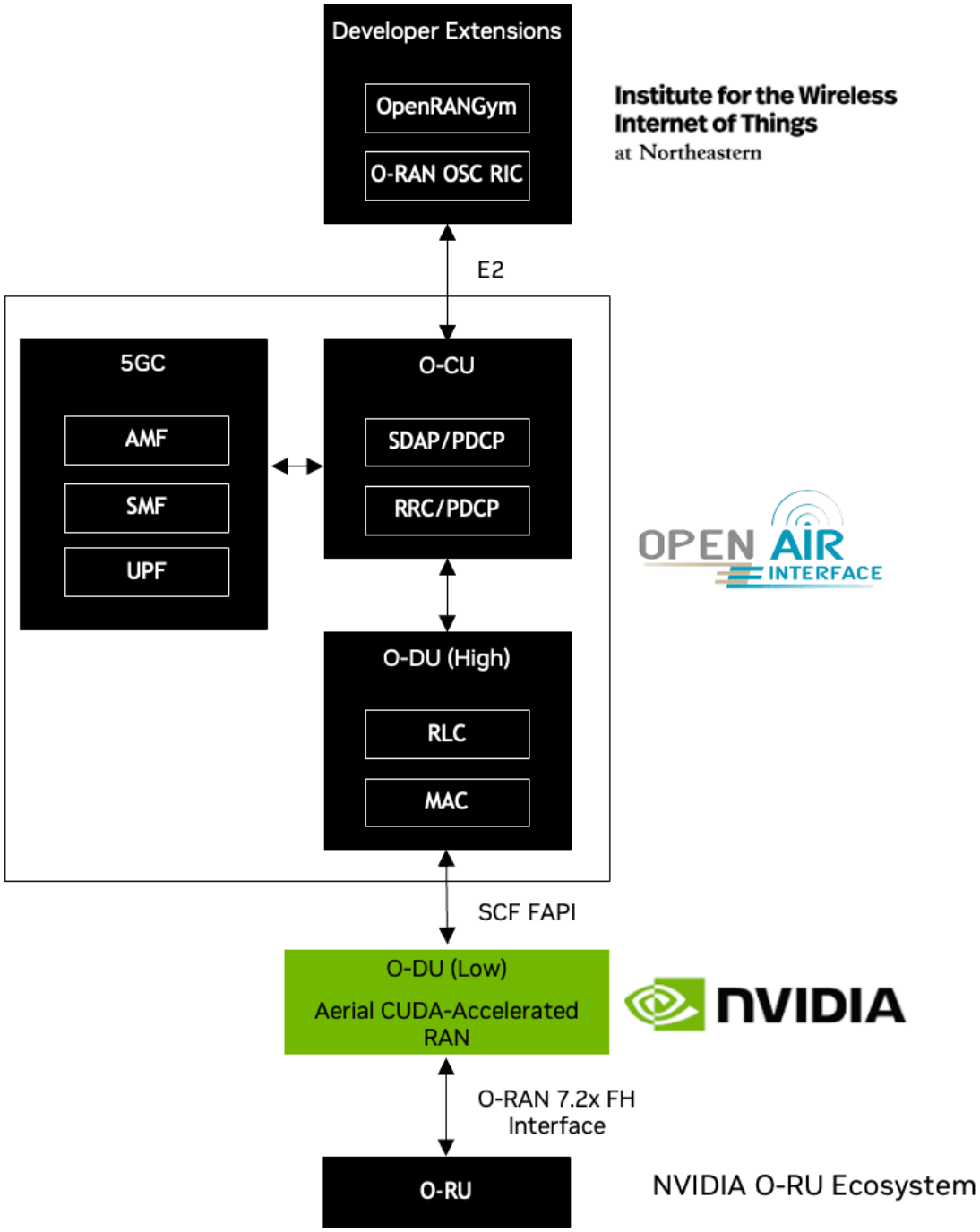
The O-RAN split-RAN concept disaggregates the RAN into multiple functional components: O-RAN Central Unit (O-CU), O-RAN Distributed Unit (O-DU) and O-RAN Radio Unit (O-RU), as shown in the figure below. These components can be deployed on different hardware and software platforms and can be interconnected using open interfaces.



ATB conforms to an O-RAN blueprint as shown in the figure and table below. The figure highlights the multi-vendor aspect of O-RAN, with O-RUs supplied from NVIDIA O-RU ecosystem partners, Layer-1 from NVIDIA Aerial-CUDA Accelerated RAN, and O-DU (high), O-CU, and 5G Core from OAI.

The *developer extension* from Northeastern University (NEU) has the following highlights:

1. OpenRAN Gym, an open toolbox for data collection and experimentation with AI in O-RAN architectures
2. Integration of the OSC (O-RAN Software Community) near real-time RIC (RAN Intelligent Controller)


Developer Extensions
OpenRANGym
O-RAN OSC RIC
Institute for the Wireless Internet of Things
at Northeastern
E2
5GC
AMF
SMF
UPF
O-CU
SDAP/PDCP
RRC/PDCP
OPEN AIR INTERFACE
O-DU (High)
RLC
MAC
SCF FAPI
O-DU (Low)
Aerial CUDA-Accelerated RAN
NVIDIA
O-RAN 7.2x FH Interface
O-RU
NVIDIA O-RU Ecosystem

Organization	Components
Northeastern	E2 interface plugin leveraging O-RAN OSC RIC and template xApps
OAI	O-DU-High (Layer 2), O-CU and 5GC
NVIDIA	O-DU Low / High PHY
Others	Handsets (Apple iPhone 14, Samsung S23), Viavi Qualsar Grandmaster, Dell FH switch, Supermicro GH200 server.

Chapter 4. Installation Guide

The following sections will guide you through the process of integrating and deploying Aerial Testbed (ATB):

Installation Step	Description
Part 1 – Procure the Hardware	Procure all the required hardware based on the published bill of materials (BOM) in this document.
Part 2 – Configure the Network Hardware	Perform setup on the required network devices.
Part 3 – Install ATB Software Components	Install ATB software components.
Part 4 – Validate the Setup	Validate the setup using bi-directional UDP.

4.1. Part 1. Procure the Hardware

Procure all the hardware listed in the following BOM.

Note

Unless a specific solution architecture differs based on use case, all components are required in a unit of 1.

5G Infra Component	Hardware Manifest		
ATB gNB (only one required)	SMC-GH: Supermicro Server ARS-11GL-NHR (Config 2)	CPU	72-core NVIDIA Grace
		GPU	GH200
		Memory	480GB LPDDR5X with ECC
		Network	BF3 NIC (x2)
	DGX Spark	CPU	20-core ARM processor (10 Cortex-X925 + 10 Cortex-A725)
		GPU	NVIDIA Blackwell Architecture (GB10)
		Memory	128 GB unified system memory
		Network	2x QSFP Network connectors (ConnectX-7)
CN[1]	x86-based or ARM-based server		
Fronthaul (FH) Switch (only one required)	Dell PowerSwitch S5248F-ON		
	Adva XG480		
	Ciena 5164		
	Cisco Nexus 9300		
	Spectrum SN3750/SN5400		
	FibroLAN Falcon RX		
GrandMaster (GM)	VIAVI Qg 2 Multi-Sync Gateway and PTP GrandMaster		
O-RUs supported	**ORU**	**Freq Band**	
	WNC 1220 (4T4R)	(indoors) 3.55GHz - 3.77GHz	
UEs supported	Samsung Galaxy S22, 23	SU-MIMO 4DL, 1UL	
Cables	Dell C2G 1m LC-LC 50/125 Duplex Multimode OM4 Fiber Cable Aqua 3ft Optical patch cable		
	NVIDIA MCP1600-C001E30N DAC Cable Ethernet 100GbE QSFP28 1m		
	Beyondtech 5m (16ft) LC UPC to LC UPC Duplex OM3 Multimode PVC (OFNR) 2.0mm Fiber Optic Patch Cable		
	CableCreation 3ft Cat5/Cat6 Ethernet Cables		
PDUs	Tripp Lite 1.4kW Single-Phase Monitored PDU with LX Platform Interface, 120V Outlets (8 5-15R), 5-15P, 12ft Cord, 1U Rack-Mount, TAA		
Transceiver	Finisar SFP-to-RJ45 Transceiver		
	Intel Ethernet SFP+SR Optics		
	Dell SFP28-25G-SR Transceiver		
Ethernet Switch	Netgear ProSafe Plus JGS524E Rackmount Switch		

Important

The following components have reached end-of-life (EOL).

- Gigabyte Edge E251-U70 Server
- A100X (will no longer be available from distributor for Aerial Testbed)

4.2. Part 2. Configure the Network Hardware

The network hardware is configured in the following steps.

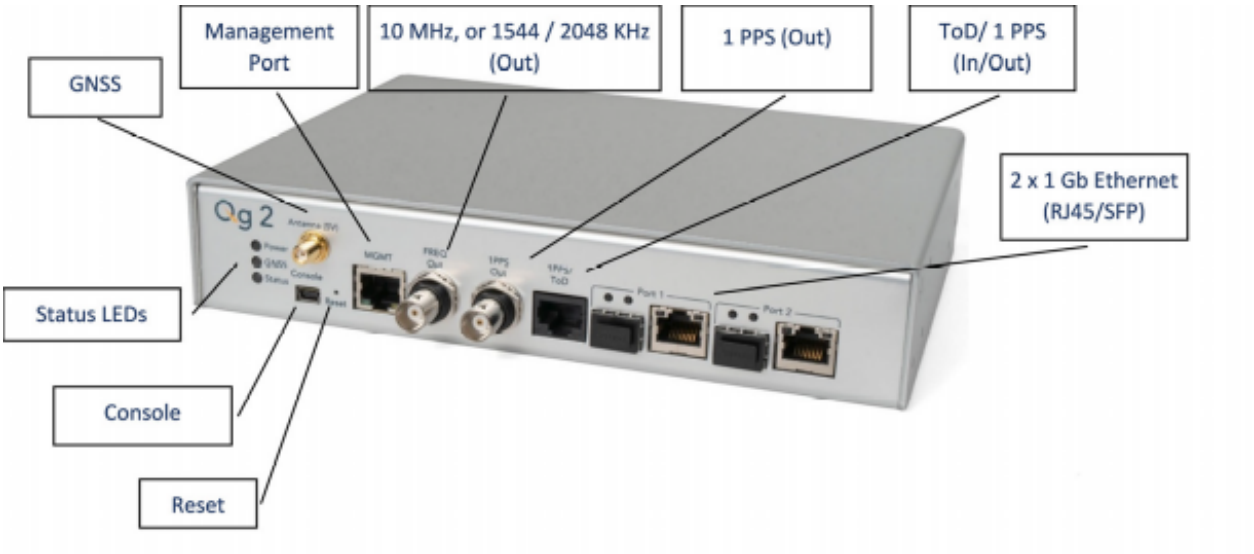
1. Set up the GrandMaster
2. Set up the switch
3. Set up the O-RU

4.2.1. Part 2.1 - Setup the VIAVI Solutions GrandMaster

The Qg 2 (picture below) is a small form factor, highly accurate Multi-Sync Gateway that provides IEEE 1588-2008 PTP Grand Master and Boundary Clock functionality. IEEE 1588-2008 PTP is also known as **PTP Version 2**. It is used for synchronizing the Aerial Testbed system.

Follow the steps in the VIAVI User Guide to configure the Qg 2.

Front Panel



Back Panel

[1] The same SMC-GH server can be used to host both gNB and ATB 5GC.

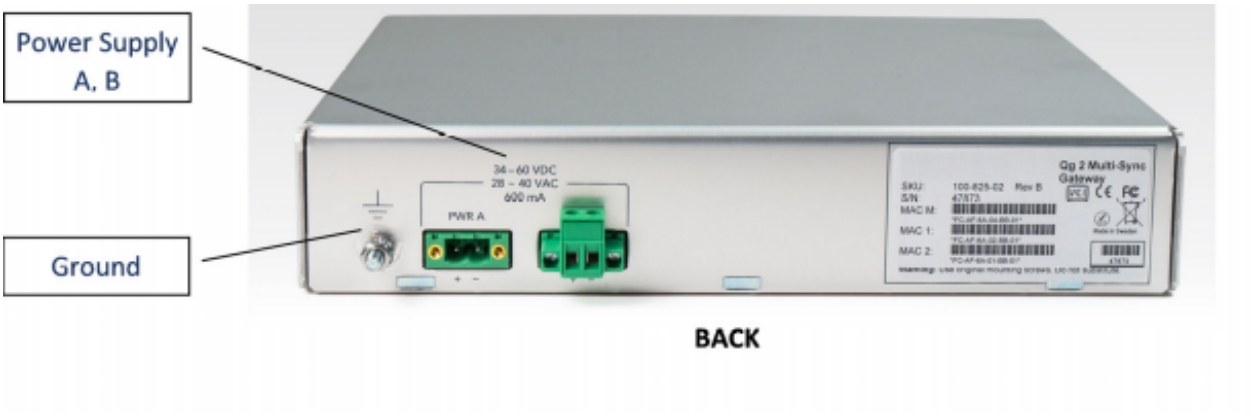


Fig. 1: Figure credit: VIAVI QG2 Multi-Sync Gateway Users Guide

4.2.2. Part 2.2 - Set up the Switch

4.2.2.1 Dell PowerSwitch S5248F-ON

The following example uses these VLAN 2 settings:

- RUs are on ports 1 and 7
- GrandMaster is on port 5
- CN is on ports 11 and 12
- gNB ports are connected to ports 49 and 51

1. Set up MGMT access to the switch (in this case 172.168.20.67):

```
OS10# configure terminal
OS10(config)#
interface mgmt1/1/1
no shutdown
no ip address dhcp
ip address 172.16.204.67/22
exit
```

2. Use SSH to access `admin@172.168.204.67`.
3. Set the speed to 10G for port groups 1 and 2.

```
OS10(config)#
port-group 1/1/1
mode Eth 10g-4x
exit
port-group 1/1/2
mode Eth 10g-4x
exit
```

4. Enable PTP on the switch.

```
OS10# configure terminal
OS10(config)#
```

(continues on next page)

(continued from previous page)

```
ptp clock boundary profile g8275.1
ptp domain 24
ptp system-time enable
!
```

5. Configure the GrandMaster port.

```
OS10(config)#
interface ethernet 1/1/5:1
no shutdown
no switchport
ip address 169.254.2.1/24
flowcontrol receive off
ptp delay-req-min-interval -4
ptp enable
ptp sync-interval -4
ptp transport layer2
exit
```

After some time, the following values will print:

```
<165>1 2023-05-09T07:49:22.625584+00:00 OS10 dn_alm 1021 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %PTP_SYSTEM_TIME_NOT_SET: System time is
↪not set. System time will be set when the clock is.
<165>1 2023-05-09T07:51:22.312557+00:00 OS10 dn_alm 1021 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %PTP_CLOCK_PHASE_LOCKED: Clock servo is
↪phase locked.
<165>1 2023-05-09T07:51:22.313081+00:00 OS10 dn_alm 1021 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %PTP_SYSTEM_TIME_UPDATE_STARTED: System
↪time update service is started. Update interval: 60 minutes.
<165>1 2023-05-09T07:51:59.334346+00:00 OS10 dn_alm 1021 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %ALM_CLOCK_UPDATE: Clock changed
↪MESSAGE=apt-daily.timer: Adding 6h 36min 18.719270s random time.
<165>1 2023-05-09T07:57:27.254181+00:00 OS10 dn_alm 1021 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %ALM_CLOCK_UPDATE: Clock changed
↪MESSAGE=apt-daily.timer: Adding 4h 31mi
```

6. Configure the Fronthaul Network Configuration by creating a VLAN.

Note

If you choose to use a different VLAN, you must modify the Aerial YAML file and O-RU configuration. C- and U-planes use the same VLAN.

Create “VLAN 2”.

```
OS10(config)#
interface vlan 2
OS10(conf-if-vl-2)#
<165>1 2023-03-16T16:51:36.458730+00:00 OS10 dn_alm 813 - - Node.1-Unit.
↪1:PRI [event], Dell EMC (OS10) %IFM_ASTATE_UP: Interface admin state up
↪:vlan2
```

(continues on next page)

(continued from previous page)

```
OS10(conf-if-vl-2)# show configuration
!
interface vlan2
no shutdown
OS10(conf-if-vl-2)# exit
```

7. Configure the RU, gNB, and CN ports.

   Interfaces that are configured to be slower than their maximum speed have a :1 appended to their name. This applies to ports in port groups 1 and 2.

```
no shutdown
switchport mode trunk
switchport trunk allowed vlan 2
mtu 8192
flowcontrol receive off
ptp enable
ptp transport layer2
ptp role timeTransmitter
exit
```

8. Check the PTP status.

```
OS10# show ptp | no-more
PTP Clock : Boundary
Clock Identity : b0:4f:13:ff:ff:46:63:5f
GrandMaster Clock Identity : fc:af:6a:ff:fe:02:bc:8d
Clock Mode : One-step
Clock Quality
Class : 135
Accuracy : <=100ns
Offset Log Scaled Variance : 65535
Domain : 24
Priority1 : 128
Priority2 : 128
Profile : G8275-1(Local-Priority:-128)
Steps Removed : 1
Mean Path Delay(ns) : 637
Offset From Master(ns) : 1
Number of Ports : 8
------------------------------------------------------------------------------
↪--
Interface State Port Identity
------------------------------------------------------------------------------
↪--
Ethernet1/1/1:1 Master b0:4f:13:ff:ff:46:63:5f:1
Ethernet1/1/3:1 Master b0:4f:13:ff:ff:46:63:5f:3
Ethernet1/1/5:1 Slave b0:4f:13:ff:ff:46:63:5f:5
Ethernet1/1/7:1 Master b0:4f:13:ff:ff:46:63:5f:8
Ethernet1/1/11 Master b0:4f:13:ff:ff:46:63:5f:4
Ethernet1/1/49 Master b0:4f:13:ff:ff:46:63:5f:9
Ethernet1/1/51 Master b0:4f:13:ff:ff:46:63:5f:10
```

(continues on next page)

(continued from previous page)

```
Ethernet1/1/54 Master b0:4f:13:ff:ff:46:63:5f:2
------------------------------------------------------------------------
↪--
Number of slave ports :1
Number of master ports :7
```

9. Save the switch configuration:

```
copy running-configuration startup-configuration
```

4.2.2.2 Ciena 5164

Use these the following documents as reference when setting up the Ciena Switch:

- `009-3407-008_(5170_10_6_Base_Advanced_Ethernet_and_OAM_Configuration)RevA.pdf`
- `009-3407-043_(5170_10_6_Synchronization_Configuration)RevisionA.pdf`

The screen shots in these documents are from a setup where the following equipment is connected to the switch:

- Port 2 (vlan2) <-> Foxconn RU
- Port 5 and 6 (n1) <-> CN
- Port 7 (n6) <-> CN
- Port 8 (n1) <-> gNB
- Port 21 (n6) <-> MEC
- Port 24 (n1) <-> gNB
- Port 35 (vlan2) <-> gNB

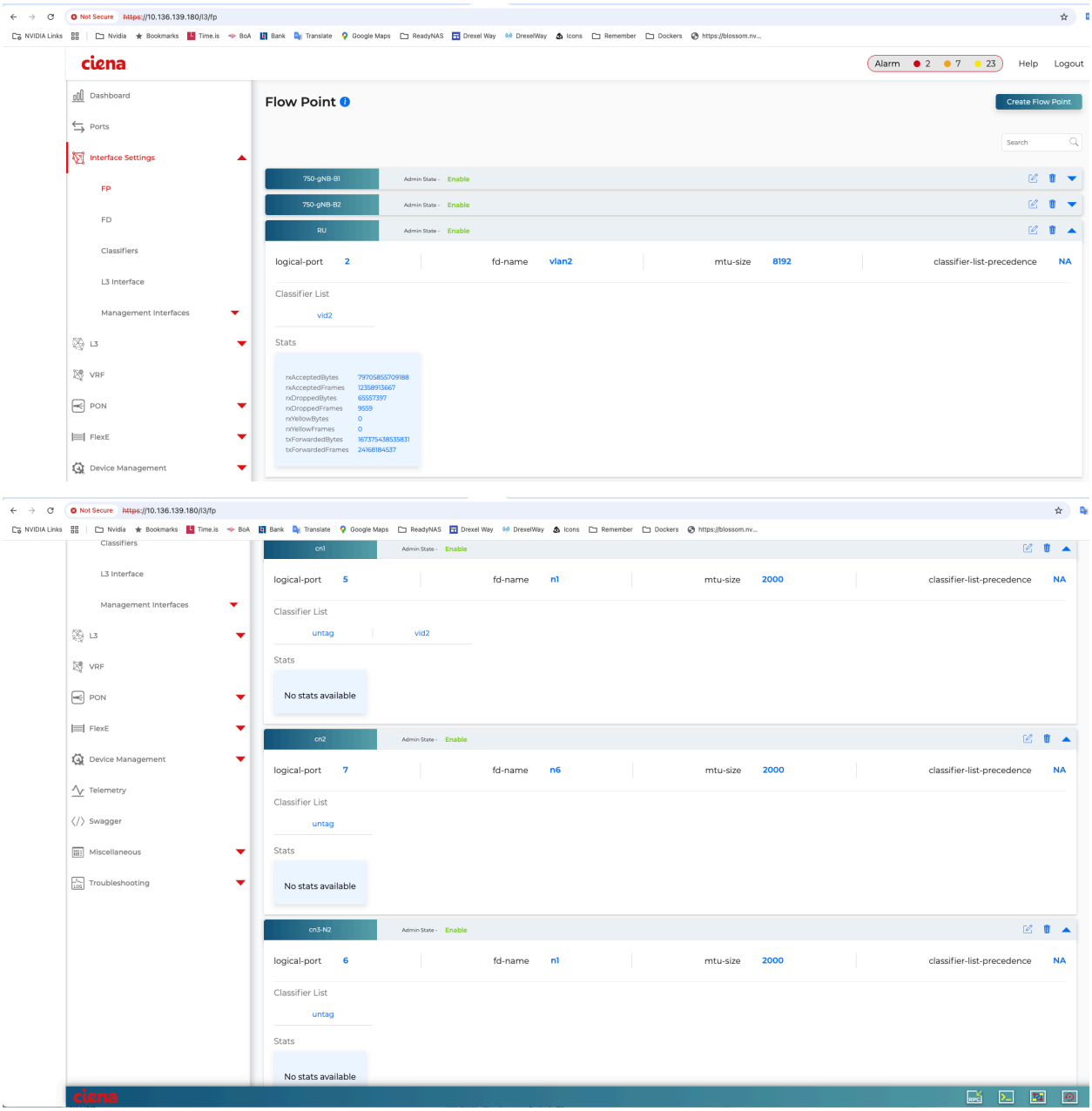
Note

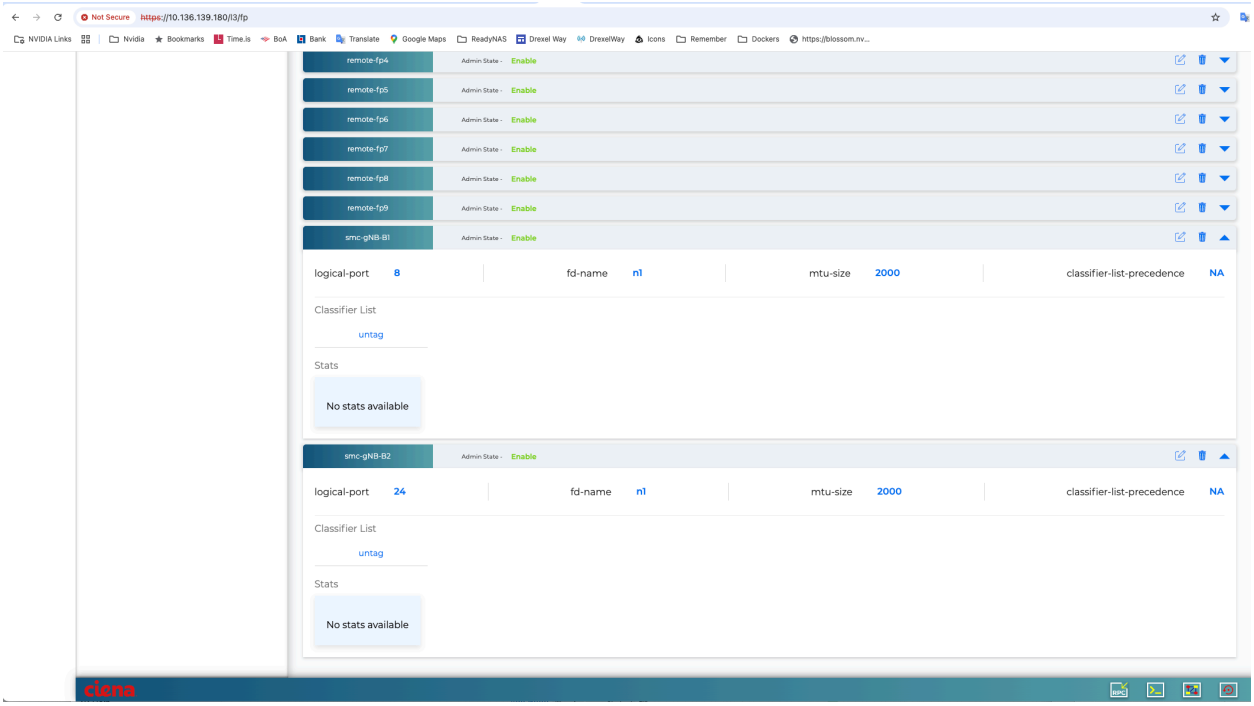
The switches in the screen shots have other equipment connected that is irrelevant.

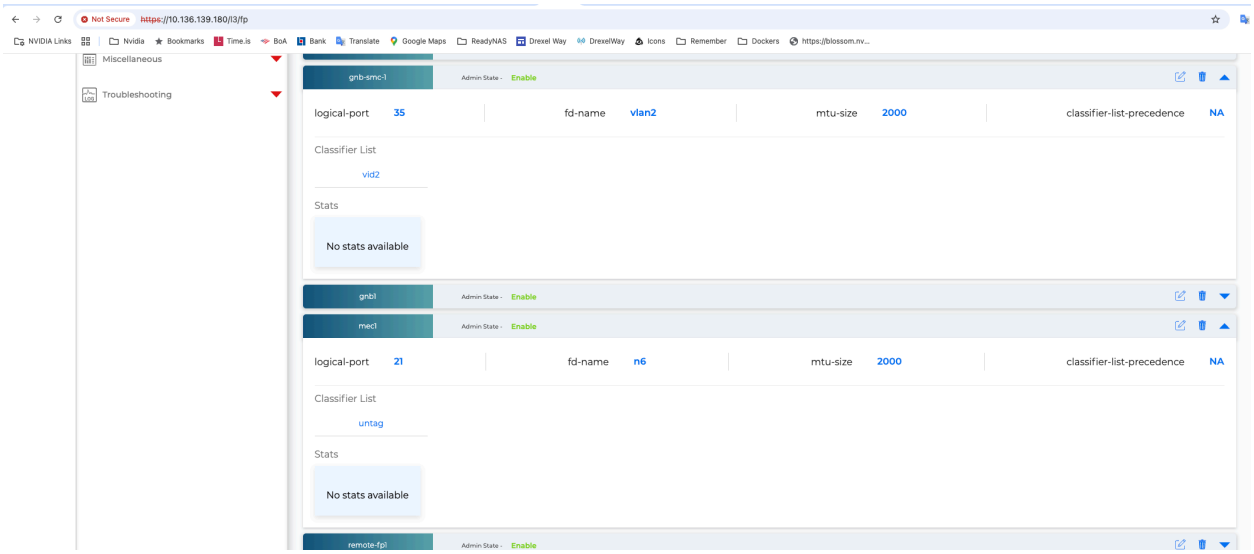
The switch has been set up so that it can be accessed both from a browser and using SSH.

```
https://<IP_ADDRESS>/dashboard/view
```

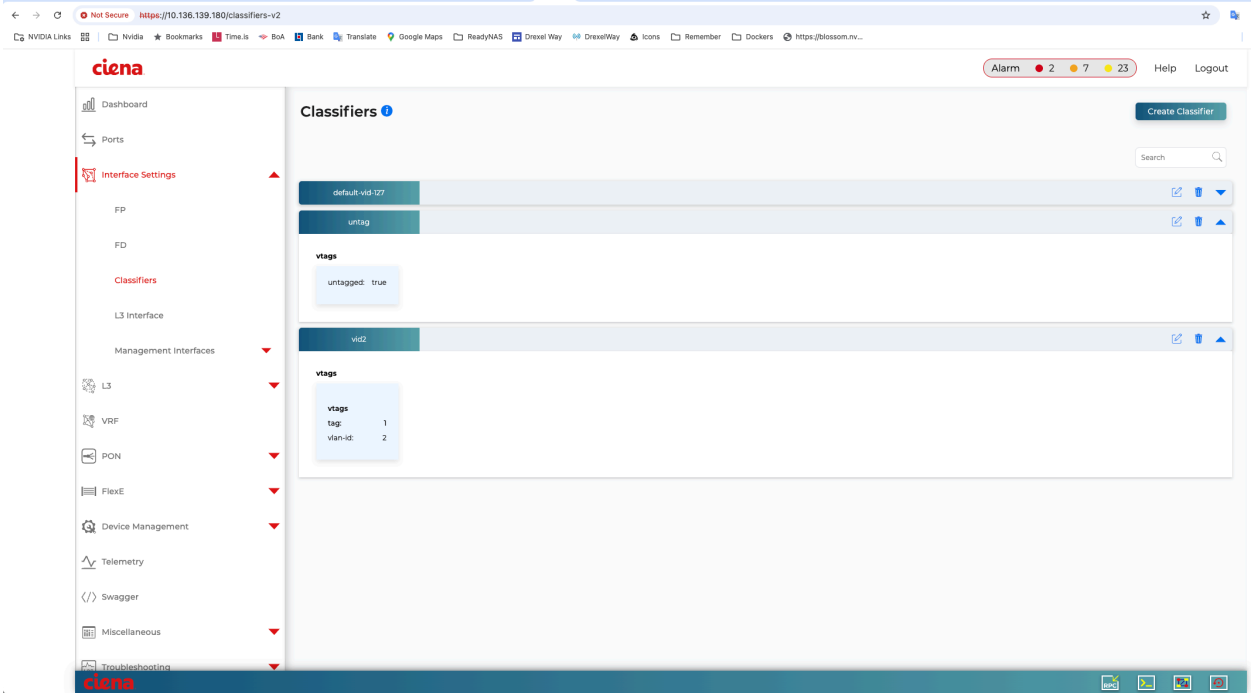
4.2.2.2.1 Create the Flow Points



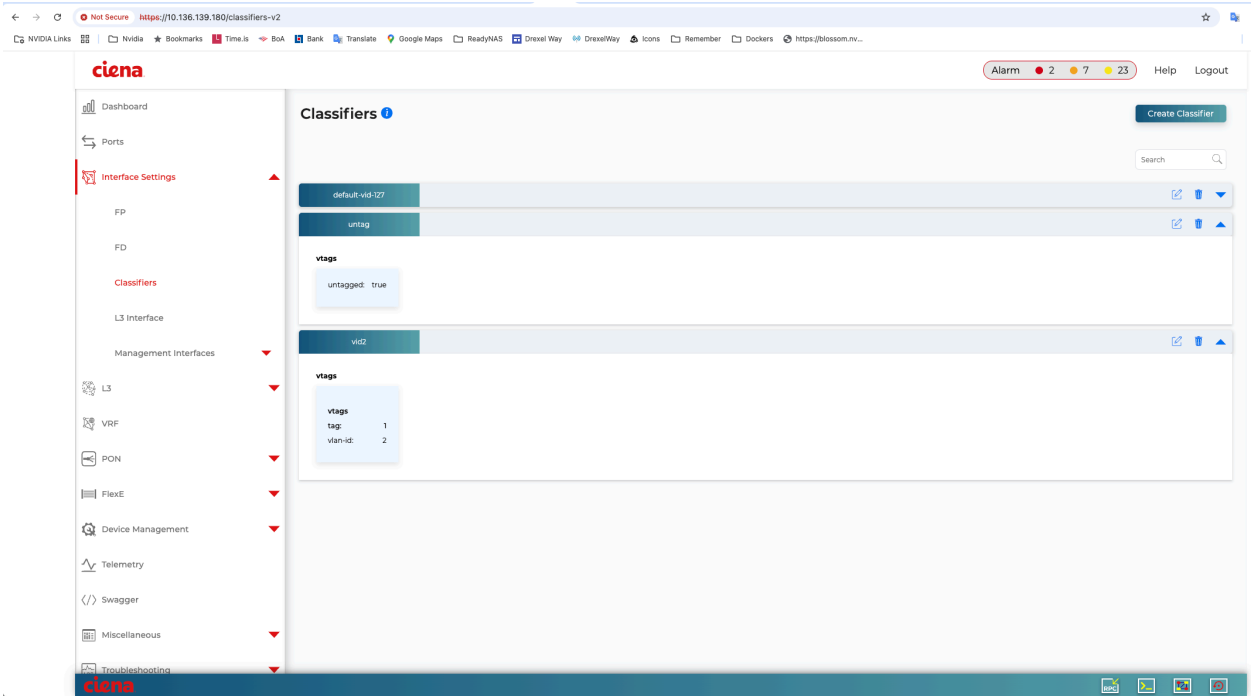

Not Secure https://10.136.139.180/l3/fp
remote-fp4 Admin State - Enable
remote-fp5 Admin State - Enable
remote-fp6 Admin State - Enable
remote-fp7 Admin State - Enable
remote-fp8 Admin State - Enable
remote-fp9 Admin State - Enable
smc-gNB-B1 Admin State - Enable
logical-port 8
fd-name n1
mtu-size 2000
classifier-list-precedence NA
Classifier List
untag
Stats
No stats available
smc-gNB-B2 Admin State - Enable
logical-port 24
fd-name n1
mtu-size 2000
classifier-list-precedence NA
Classifier List
untag
Stats
No stats available
ciena


Not Secure https://10.136.139.180/l3/fp
Miscellaneous
Troubleshooting
gnb-smc-1 Admin State - Enable
logical-port 35
fd-name vlan2
mtu-size 2000
classifier-list-precedence NA
Classifier List
vid2
Stats
No stats available
gnb1 Admin State - Enable
mec1 Admin State - Enable
logical-port 21
fd-name n6
mtu-size 2000
classifier-list-precedence NA
Classifier List
untag
Stats
No stats available
remote-fp1 Admin State - Enable

4.2.2.2.2 Create the Classifiers



4.2.2.2.3 Create the Forwarding Domains



4.2.2.2.4 Setup PTP

Use SSH to connect to the switch (using the IP address of switch).

Enter Config mode, then add the ports to ptp:

```
5164-2nd> config

user@5164-2nd# sync output-references ptp-output-reference ptp_out_2
 ↪interface 2

user@5164-2nd# sync output-references ptp-output-reference ptp-out-35
 ↪interface 35

save

5164-2nd> show sync ptp

5164-2nd> show sync ptp output-references

+------------- SYNC PTP OUTPUT REFERENCES -------------+
| Name       | Interface | Oper State | PTP Port State |
+------------+-----------+------------+----------------+
| ptp_out_2  |         2 |     Up     |     Master     |
| ptp-out-35 |        35 |     Up     |     Master     |
+------------+-----------+------------+----------------+
```

4.2.2.2.5 Setup GPS

Use SSH to connect to the switch (using the IP address of switch).

Configure the GPS:

```
5164-2nd> show sync gnss antenna-inputs

+---------------------------- SYNC GNSS ANTENNA INPUT -----------------------
 ↪-----+
| Name    | Pri | Override Pri | Forced QL | Oper State | Antenna Signal
 ↪Condition |
+---------+-----+--------------+-----------+------------+--------------------
 ↪-----+
| GNSS_in |   - |            - |  QL-PRTC  |   Locked   |          Normal
 ↪     |
+---------+-----+--------------+-----------+------------+--------------------
 ↪-----+

5164-2nd> show sync gnss satellites

+-- SYNC GNSS SATELLITES --+
| PRN | Acquired | SV Type |
+-----+----------+---------+
|  10 | Acquired |   GPS   |
|  23 | Acquired |   GPS   |
|  32 | Acquired |   GPS   |
```

(continues on next page)

(continued from previous page)

```
|  8  | Acquired |   GPS   |
|  21 | Acquired |   GPS   |
|  2  | Acquired |   GPS   |
|  31 | Acquired |   GPS   |
|  27 | Acquired |   GPS   |
+-----+----------+---------+

5164-2nd> show sync gnss almanac

+------ SYNC GNSS ALMANAC ------+
| PRN | SV Health | Week Number |
+-----+-----------+-------------+
|  31 |     0     |     220     |
|  27 |     0     |     220     |
|  32 |     0     |     220     |
|  21 |     0     |     222     |
|  2  |     0     |     220     |
|  8  |     0     |     220     |
|  10 |     0     |     220     |
|  23 |     0     |     220     |
+-----+-----------+-------------+
```

4.2.2.3 FibroLAN Falcon RX

Although the FibroLAN switch has not been qualified in the NVIDIA lab, OAI labs incorporate the following configuration and switch for interoperability.

To get started, follow the FibroLAN Falcon R Class Quick Guide Getting Started.

In this setup, the VIAVI GrandMaster is connected to port 4, the Aerial cuBB to port 17, and the O-RU to port 16 (C/U plane) and port 15 (S/M plane). You can ignore all other ports in the figures [A][B] below.

4.2.2.3.1 VLAN Setup

The following assumes that the VLAN tag is 2 for both the control plane and the user plane of the O-RAN CU plane. VLAN tag 80 is used for everything else.

Open the configuration page of the FibroLAN switch, then go to **Configuration > VLANs**. Port 4 (the VIAVI GrandMaster) needs to be set to “Access” mode, with the port VLAN set to 80.

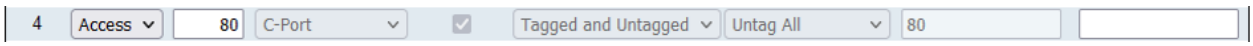


Fig. 2: Figure A - VLAN Setup

Use the same configuration for port 15 (RU S/M plane).

Configure ports 16 and 17 as follows:

- **Mode**: “Trunk”
- **Port**: VLAN 80
- **Untag Port VLAN**
- **Allowed VLANs**: 2, 80

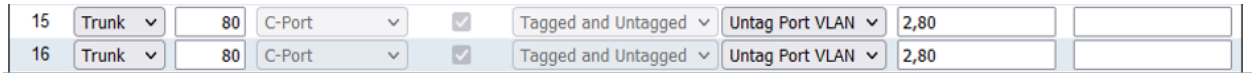
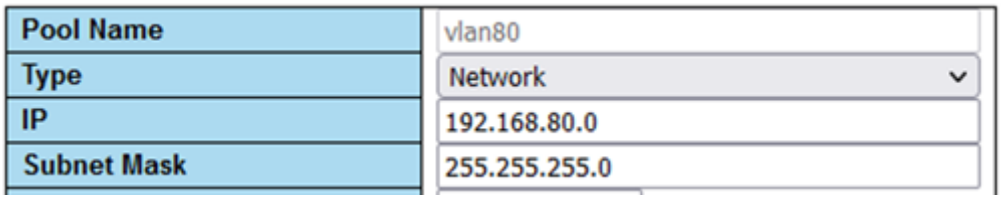


Fig. 3: Figure B - VLAN Setup

4.2.2.3.2 DHCP Setup

The ORU M-plane requires you to set up a DHCP server. Go to **Configuration > DHCP > Server > Pool** and create a new DHCP server with the following settings:



4.2.2.3.3 PTP Setup on gNB

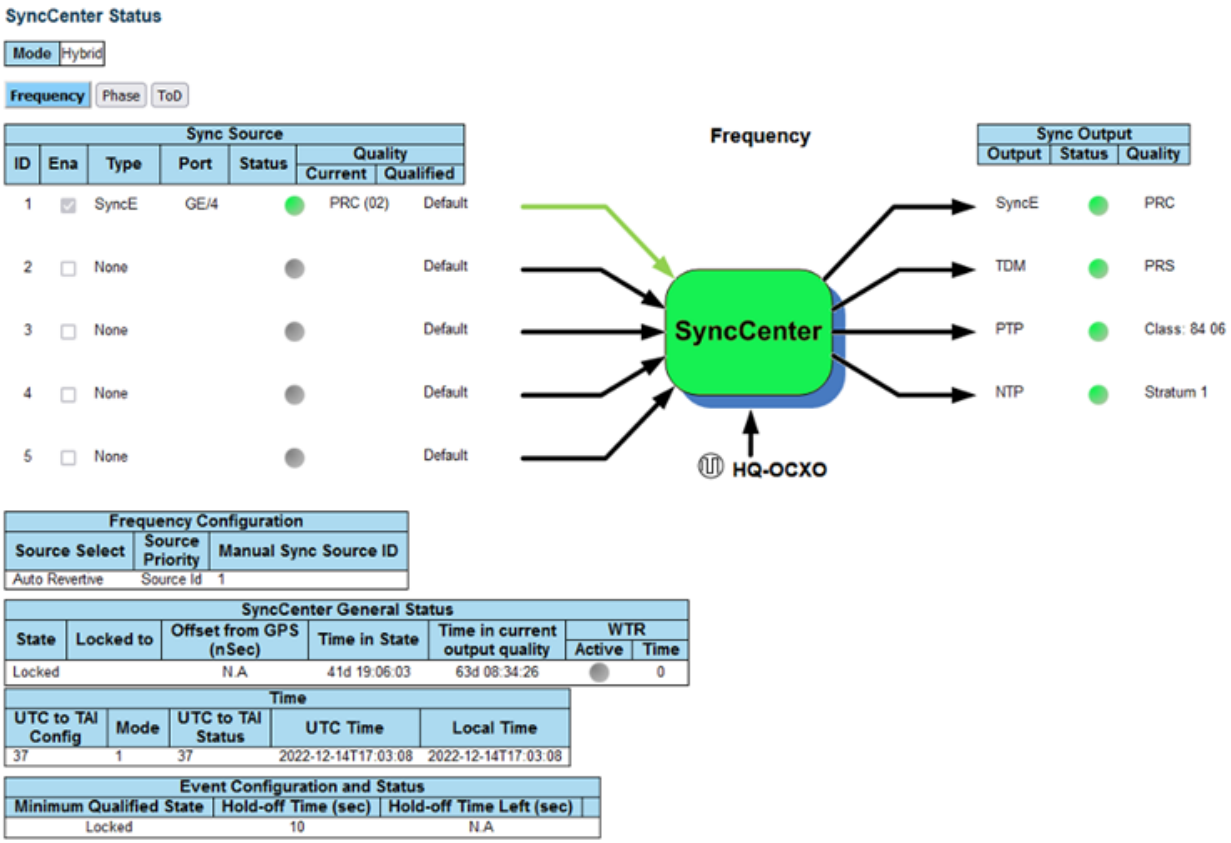
For the PTP setup, follow the Fibrolan PTP Boundary Clock Configuration guide and use the following settings:

- Device Type: “Ord-Bound”
- Profile: “G8275.1”
- Clock domain: 24
- VLAN: 80

Also make sure you enable the used ports (in this case, 4, 15, 16, and 17).

Hybrid mode is recommended as the sync mode.

If everything is configured correctly, the SyncCenter should show green.



4.2.3. Part 2.3 - Set up the Foxconn O-RU

Note

Foxconn O-RUs are no longer supported. These instructions are included for previously installed O-RUs only.

4.2.3.1 O-RU M-Plane Setup

1. Add the following to the bottom of `/etc/profile` and comment out the line with `set_qse.sh` if it already exists. Set the interface initially to `eth0` for firmware version 1, and to `qse-eth` after upgrading to firmware version 2 or greater.

```
interface=eth0
vlanid=2
ipLastOctet=20

ip link add link ${interface} name ${interface}.$vlanid type vlan id
↪$vlanid
ip addr flush dev ${interface}
ip addr add 169.254.0.0/24 dev ${interface}
ip addr add 169.254.1.${ipLastOctet}/24 dev ${interface}.$vlanid
ip link set up ${interface}.$vlanid
```

2. Reboot the O-RU using the command `./reboot.sh` and check the network configuration:

```
# ip r
169.254.1.0/24 dev eth0.2 src 169.254.1.20
```

4.2.3.2 Update O-RU Configuration

Note

If you are using the CBRS O-RU, refer to the note below for the modified configuration.

1. Update the O-RU configuration in `/home/root/test/RRHconfig_xran.xml`.

```
root@arria10:~/test# grep -v '<!-' RRHconfig_xran.xml
RRH_DST_MAC_ADDR = 08:c0:eb:71:e7:d4 # To match fronthaul interface of DU
RRH_SRC_MAC_ADDR = 6C:AD:AD:00:04:6C # To match qse-eth of RU
RRH_EN_EAXC_ID = 0
RRH_EAXC_ID_TYPE1 = 0x0, 0x1, 0x2, 0x3
RRH_EAXC_ID_TYPE3 = 0x8, 0x9, 0xA, 0xB
RRH_EN_SPC = 1
RRH_RRH_LTE_OR_NR = 1
RRH_TRX_EN_BIT_MASK = 0x0f
RRH_RF_EN_BIT_MASK = 0x0f
RRH_CMPR_HDR_PRESENT = 0
RRH_CMPR_TYPE = 1, 1
RRH_CMPR_BIT_LENGTH = 9, 9
RRH_UL_INIT_SYM_ID = 0
RRH_TX_TRUNC_BITS = 4
RRH_RX_TRUNC_BITS = 4
RRH_MAX_PRB = 273
RRH_C_PLANE_VLAN_TAG = 0x0002 #To match vlan id set in cuphycontroller
↪yaml file
RRH_U_PLANE_VLAN_TAG = 0x0002 #To match vlan id set in cuphycontroller
↪yaml file
RRH_SLOT_TICKS_IN_SEC = 2000
RRH_SLOT_PERIOD_IN_SAMPLE = 61440
RRH_LO_FREQUENCY_KHZ = 3750000, 0
RRH_TX_POWER = 24, 24
RRH_TX_ATTENUATION = 12.0, 12.0, 12.0, 12.0
RRH_RX_ATTENUATION = 0.0, 0.0, 0.0, 0.0
RRH_BB_GENERAL_CTRL = 0x0, 0x0, 0x0, 0x0
RRH_RF_GENERAL_CTRL = 0x3, 0x1, 0x0, 0x0
RRH_PTPV2_GRAND_MASTER_MODE = 3
RRH_PTPV2_JITTER_LEVEL = 0
RRH_PTPV2_VLAN_ID = 0
RRH_PTPV2_IP_MODE = 4
RRH_PTPV2_GRAND_MASTER_IP = 192.167.27.150
RRH_PTPV2_SUB_DOMAIN_NUM = 24
RRH_PTPV2_ACCEPTED_CLOCK_CLASS = 135
RRH_TRACE_PERIOD = 10
RRH_DL_IQ_SCALING = 0x1001
RRH_CFR_PEAK_THRESHOLD = 0.5
```

Note

The above configuration was taken from an ORU running firmware 3.1.15.

Note

If you're using the CBRS O-RU, the above parameters should be modified as follows:

`RRH_LO_FREQUENCY_KHZ = 3649140, 0`

2. Reboot O-RU.

```
cd /home/root/test/

./reboot
```

3. Run the following to enable the configuration:

```
cd /home/root/test/

./init_rrh_config_enable_cuplane
```

4. To see the ORU status, run the following script.

```
cd /home/root/test/

./chk_con.sh
```

4.2.4. Part 2.4 - Set up the WNC O-RU

When the the O-RU starts up, it will have a baseline configuration. Access the O-RU using the following command: `ssh root@192.168.2.1` (no password is required). The radio interfaces will be "shutdown".

```
ru# show running-config
...
interface eth1
  no shutdown
  mac-address e8:c7:cf:ac:58:32
  l2-mtu 9600
  ip address 192.168.2.1/24
  sfp rs0-high
  no sfp rs1-high
  ip dhcp client vendor-class-identifier o-ran-ru2/WNC/R1220-077L
  sub-interface eth1.mplane encapsulation vlan 100
  sub-interface eth1.ecpri encapsulation vlan 564
  exit
!
...
radio 1
  shutdown
  center-freq 3750000
```

(continues on next page)

(continued from previous page)

```
  transmit-power 24
  lna shutdown
```

Then SSH again as shown above. Start the radio as shown below. Look for the “no shutdown” message.

```
============================
WNC O-RU Command Line System
============================
ru# radio enable
ru# show running-config
!
hostname ru
!
...
radio 1
  no shutdown
  center-freq 3750000
  transmit-power 24
  no lna shutdown
```

Check PTP:

```
ru# show ptp clock
PTP Clock Information:
  PTP Device Type                  : slave clock
  Clock Identify                   : e8c7cf.fffe.ac5832
  Profile                          : g.8275.1
  Clock Domain                     : 24
  Number of PTP ports              : 1
  Priority1                        : 128
  Priority2                        : 128
  Clock Quality                    :
      Class                        : 255
      Accuracy                     : 0xfe
      Offset (log variance)        : 0xffff
  Offset From Master(ns)           : -4.0
  Mean Path Delay(ns)              : 206.0
  Steps Removed                    : 2
  S-plane State                    : locked
OK
```

PCP is fixed at 7 with the WNC radio and cannot be changed. This must be changed in the *cuphycontroller.yaml* file.

Changing VLAN can be done as follows:

1. Tell the radio to use a different interface for eCPRI transport
2. Delete the ecpri interface with old vlan
3. Create the eCPRI interface with new vlan
4. Change the radio to use that eth1.ecpri interface again
5. Or see section 2.2.7 of the WNC manual.

```
ru# config
OK
ru(config)# radio 1
OK
ru(conf-rf 1)# transport interface eth1.mplane
OK
ru(conf-rf 1)# exit
OK
ru(config)# interface eth1
OK
ru(conf-if eth1)# no sub-interface eth1.ecpri
OK
ru(conf-if eth1)# sub-interface eth1.ecpri encapsulation vlan 2
OK
ru(conf-if eth1)# Listening on interface eth1.ecpri for CFM frames

ru(conf-if eth1)# exit
OK
ru(config)# radio 1
OK
ru(conf-rf 1)# transport interface eth1.ecpri
OK
```

4.2.4.1 Reference Configuration

For refence, Aerial Testbed has been verified with the following radio configuration.

```
radio 1
  no shutdown
  center-freq 3750000
  transmit-power 24
  no lna shutdown
  lna low-gain
  eaxc-id downlink 0x0000
  eaxc-id uplink 0x0000
  eaxc-id prach 0x0004
  transport interface eth1.ecpri
  transport peer-mac 9c:63:c0:f4:26:d2
  phase-compensation
  bandwidth 100
  sub-carrier 30
  max-scs 30
  transmit-power-scale 0.0
  compress tx static bfp iq-bitwidth 9
  compress rx static bfp iq-bitwidth 9
  compress prach static bfp iq-bitwidth 9
  compress oran-compliant
  downlink-radio-frame-offset 0
  downlink-sfn-offset 0
  ul-gain-correction 0.0000
  digital-power-scaling o-ran
  fs-offset tx 0
```

(continues on next page)

(continued from previous page)

```
  fs-offset rx 0
  fs-offset prach 0
  exit
!
```

4.3. Part 3. Install Aerial Testbed Software Components

4.3.1. Automated installation of Aerial CUDA-Accelerated RAN

Aerial CUDA-Accelerated RAN can be installed by following the Aerial CUDA-Accelerated RAN Installation Guide.

For your convenience, the aerial-cuda-accelerated-ran repository includes the NVIDIA Aerial SDK Installation script, which automates the installation process.

From a fresh flash of DGX Spark, or from a fresh Ubuntu Server 22.04.5 installation, after the clone above, install using the scripts under `cubb_scripts/install/`. It is recommended to run the installation in a tmux session so the installation can continue in the case of network issues. With the below commands, the session can be reattached with `tmux attach -t aerial-cuda-accelerated-ran`.

```
git clone --recurse-submodules https://github.com/NVIDIA/aerial-cuda-
↪accelerated-ran.git ~/aerial-cuda-accelerated-ran
cd ~/aerial-cuda-accelerated-ran/cubb_scripts/install/
sudo apt install make tmux git-lfs && git lfs pull
tmux new-session -s "aerial-cuda-accelerated-ran" -n "installation"
make all && sudo reboot
RU_MAC=<your RU MAC address> make all
```

On Ubuntu Server 22.04.5, the same sequence applies, but perform a **cold power cycle** after the second `make all` if the installation shows:

```
INFO[MISC]: NIC Firmware reset is not supported. Host power cycle is required
```

Then continue the installation by running `make all` again.

If errors occur or the host reboots before installation finishes, resume from the completed step by running `make all` again from `~/aerial-cuda-accelerated-ran/cubb_scripts/install/`.

Installation steps can be shown by prepending `DRY_RUN=1` to the `make all` command.

At this point you should have a running Aerial CUDA-Accelerated RAN installation including the L1 and OAI L2 + CN.

4.3.2. Manual Installation of Aerial CUDA-Accelerated RAN

After following Aerial CUDA-Accelerated RAN Installation Guide these additional steps are required.

4.3.2.1 Verify Inbound PTP Packets

Typically, you should see packets with `ethertype 0x88f7` on the selected interface.

```
sudo tcpdump -i aerial00 -c 5 | grep ethertype
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on aerial00, link-type EN10MB (Ethernet), capture size 262144 bytes
13:27:41.291503 48:b0:2d:63:83:ac (oui Unknown) > 01:1b:19:00:00:00 (oui
↪Unknown), ethertype Unknown (0x88f7), length 60:
13:27:41.291503 48:b0:2d:63:83:ac (oui Unknown) > 01:1b:19:00:00:00 (oui
↪Unknown), ethertype Unknown (0x88f7), length 60:
13:27:41.296727 c4:5a:b1:14:1a:c6 (oui Unknown) > 01:1b:19:00:00:00 (oui
↪Unknown), ethertype Unknown (0x88f7), length 78:
13:27:41.296784 c4:5a:b1:14:1a:c6 (oui Unknown) > 01:1b:19:00:00:00 (oui
↪Unknown), ethertype Unknown (0x88f7), length 60:
13:27:41.306316 08:c0:eb:71:e7:d5 (oui Unknown) > 01:1b:19:00:00:00 (oui
↪Unknown), ethertype Unknown (0x88f7), length 58:
```

4.3.2.2 Configure VLAN and IP Address on the gNB Server

4.3.2.2.1 Access a Foxconn RU

To be able to access a Foxconn RU over ssh, create a VLAN and IP address on the gNB server:

```
sudo ip link add link aerial00 name aerial00.2 type vlan id 2
sudo ip addr add 169.254.1.169/24 dev aerial00.2
sudo ip link set up aerial00.2
```

They can also be appended to (`/usr/local/bin/nvidia.sh`) without `sudo` so they are automatically run on reboot.

4.3.2.2.2 Access a WNC RU

Create (`/etc/netplan/wnc-access-setup.yaml`) and change the permissions:

```
sudo tee /etc/netplan/wnc-access-setup.yaml <<'EOF'
network:
  version: 2
  ethernets:
    aerial00:
      addresses:
      - 192.168.2.169/24
EOF
sudo chmod 600 /etc/netplan/wnc-access-setup.yaml
sudo netplan apply
```

You may need to reconnect to the server after this step.

4.3.2.3 Start the Aerial cuBB Container and Build the Source Code

Execute the `/cuPHY-CP/container/run_aerial.sh` script to set up cuBB. This script downloads the image and starts the container; it will then mount the local cuBB source code in the container and build it. Changes to the L1 script are stored locally.

```
# Run on host: start a docker container with the script below
cd ~/aerial-cuda-accelerated-ran
./cuPHY-CP/container/run_aerial.sh
```

Follow the Aerial cuBB documentation to build and run the cuphycontroller. The following instructions are for building and setting up the environment for running cuphycontroller. The following commands must be run from inside the cuBB container.

```
aerial@c_aerial_user:/opt/nvidia/cuBB$ ${cuBB_SDK}/testBenches/phase4_test_
↪scripts/build_aerial_sdk.sh --preset 10_02 -- -DSCF_FAPI_10_04_SRS=ON
```

4.3.2.4 gNB Configuration Files

OAI L2 configuration used in the default setup is located here.

L1 `cuphycontroller` configuration files are maintained in `cuPHY-CP/cuphycontroller/config/`.

For ATB 1.0, the setup has been validated with the following configuration files:

- cuphycontroller_P5G_WNC_GH.yaml
- cuphycontroller_P5G_WNC_DGX.yaml

Configuration files are selected automatically based on server type by `run_l1.sh`.

There is also a configuration file for Foxconn RU on Grace Hopper, which can be specified as the first argument to `run_l1.sh`, eg: `./run_l1.sh P5G_FXN_GH`.

- cuphycontroller_P5G_FXN_GH.yaml

In all configurations `dst_mac_addr` must match your RU's MAC address.

Before running the cuphycontroller, edit the `cuphycontroller_<xyz>.yaml` configuration file to point to the correct MAC address of the O-RU and the correct PCIe address of the FH interface on the gNB. Determine whether the NIC address is correct.

```
sed -i "s/ nic:.*/ nic: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_<xyz>.yaml
sed -i "s/ dst_mac_addr:.*/ dst_mac_addr: 6c:ad:ad:00:02:02/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/cuphycontroller_<xyz>.yaml
sed -i "s/ vlan:.*/ vlan: 2/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_<xyz>.yaml
```

When the build is done and the configuration files are updated, exit the container. All changes will be stored locally, so you don't need to commit the changes to a new docker image.

4.3.2.5 Creating the NVIPC Source Code Package

In the latest release, NVIPC is no longer included in the OAI repository. You must copy the source package from the Aerial cuBB SDK and add it to the OAI build files. Execute the following to create the `nvipc_src.<data>.tar.gz` file.

```
cd ~/aerial-cuda-accelerated-ran
./cuPHY-CP/gt_common_libs/pack_nvipc.sh
cp ./cuPHY-CP/gt_common_libs/nvipc_src.$(date +%Y.%m.%d).tar.gz ~/
↪openairinterface5g
```

This will create and copy the `nvipc_src.<data>.tar.gz` archive that is required to build OAI L2+ for Aerial. Instructions can also be found in the Aerial FAPI tutorial.

4.3.2.6 Build gNB Docker Image

In ATB 1.0, the OAI image is built in two steps. Instructions can be found in the Aerial FAPI tutorial.

Note

When building a Docker image, the files are copied from the filesystem into the image. After you build the image, you must make changes to the configuration inside the container.

4.3.2.7 Build OAI L2 image

Clone the OpenAirInterface5G repository.

```
git clone --branch ATB1.0_integration https://gitlab.eurecom.fr/oai/
↪openairinterface5g.git ~/openairinterface5g
cd openairinterface5g
```

Use the below commands to build the OAI L2 code:

```
cd ~/openairinterface5g/
docker build . -f docker/Dockerfile.base.ubuntu --tag ran-base:latest
docker build . -f docker/Dockerfile.gNB.aerial.ubuntu --tag oai-gnb-
↪aerial:latest
```

This will create two images:

- `ran-base:latest` includes the environment to build the OAI source code. This will be used when building the `oai-gnb-aerial:latest` image.
- `oai-gnb-aerial:latest` will be used later in the Docker Compose script to create the OAI L2 container.

4.3.2.8 Run gNB with Docker Compose

ATB 1.0 includes a Docker Compose configuration for running gNB software. Refer to the OAI instructions on how to run with Docker Compose. The Docker Compose configuration for SMC-GH servers is located in the following file path:

```
~/openairinterface5g/ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml
```

Docker Compose will start containers running cuBB and OAI L2+. The Docker Compose script includes an entry-point script for cuBB. The script points at the `cuphycontroller_xxx.yaml` configuration that you want to run. The script will determine the server used (only SMC-GH and DGX Spark have been tested) and use the correct `cuphycontroller_xxx.yaml` file. If you want to use a custom configuration, you can also supply that as an argument in the Docker Compose file. This script is located in the following file path:

```
~/openairinterface5g/ci-scripts/yaml_files/sa_gnb_aerial/aerial_l1_entrypoint.
↪sh
```

Before running the Docker Compose script, you need to create a local `.env` file to set the tag of the local OAI L2 image.

```
echo 'REGISTRY=""' > .env
echo 'TAG="latest"' >> .env
```

The tag will set in the following location of the `docker-compose.yaml` file.

```
image: ${REGISTRY-oaisoftwarealliance/}${GNB_IMG:-oai-gnb-aerial}:${TAG:-
↪develop}
```

You can now run ATB with the below commands.

```
cd ~/openairinterface5g/ci-scripts/yaml_files/sa_gnb_aerial
docker compose up -d
# console of cuBB
docker logs -f nv-cubb
# console of oai
docker logs -f oai-gnb-aerial
```

After following the instructions, you should have the following images:

```
$ docker image ls
REPOSITORY
↪    TAG                                       IMAGE ID       CREATED
↪SIZE
oai-gnb-aerial
↪    latest                                    1f382f5c8962   6 days ago
↪514MB
ran-base
↪    latest                                    5ed95a3c6ad5   6 days ago
↪2.43GB
nvcr.io/nvidia/aerial/aerial-cuda-accelerated-ran
↪    25-3-cubb                                 dfac27c8c511   12 days ago
↪27GB
```

4.3.2.9 Example Screenshot of Starting CN5G

```
aerial@e200-smc-11:~/openairinterface5g/doc/tutorial_resources/oai-cn5g$
↪docker compose up -d
[+] Running 11/11
✓ Network oai-cn5g-public-net  Created                0.1s
✓ Container oai-ext-dn         Started                0.6s
✓ Container mysql              Started                0.4s
✓ Container oai-nrf            Started                0.5s
✓ Container asterisk-ims       Started                0.5s
✓ Container oai-udr            Started                0.7s
✓ Container oai-udm            Started                0.9s
✓ Container oai-ausf           Started                1.1s
✓ Container oai-amf            Started                1.3s
✓ Container oai-smf            Started                1.4s
✓ Container oai-upf            Started                1.6s
```

4.3.2.10 Example Screenshot of Running CN5G

```
aerial@e200-smc-11:~$ docker ps
CONTAINER ID   IMAGE                                     COMMAND
↪  CREATED        STATUS               PORTS
```

(continues on next page)

(continued from previous page)

```
↪                  NAMES
21f4d9fc7b7a   oaisoftwarealliance/oai-upf:develop     "sh /openair-upf/bin…"
↪   5 days ago     Up 5 days (healthy)     2152/udp, 8805/udp, 5342-5344/tcp
↪                    oai-upf
dec8c3999718   oaisoftwarealliance/oai-smf:develop     "/openair-smf/bin/oa…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 5342-5344/tcp, 8080/tcp,
↪9090/tcp, 8805/udp   oai-smf
5b39b97eca1f   oaisoftwarealliance/oai-amf:develop     "/openair-amf/bin/oa…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 5342-5344/tcp, 8080/tcp,
↪9090/tcp, 38412/sctp oai-amf
4395198c8f63   oaisoftwarealliance/oai-ausf:develop    "/openair-ausf/bin/o…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 5342-5344/tcp, 8080/tcp
↪                    oai-ausf
198740927b35   oaisoftwarealliance/oai-udm:develop     "/openair-udm/bin/oa…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 5342-5344/tcp, 8080/tcp
↪                    oai-udm
ba7c80dacdfc   oaisoftwarealliance/oai-udr:develop     "/openair-udr/bin/oa…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 8080/tcp
↪                    oai-udr
f0b31e825e61   oaisoftwarealliance/oai-nrf:develop     "/openair-nrf/bin/oa…"
↪   5 days ago     Up 5 days (healthy)     80/tcp, 5342-5344/tcp, 8080/tcp
↪                    oai-nrf
27be7c0d2e6a   oaisoftwarealliance/trf-gen-cn5g:latest "/bin/bash -c ' ip r…"
↪   5 days ago     Up 5 days (healthy)
↪                    oai-ext-dn
ddbbd35aa2d9   mysql:8.0                               "docker-entrypoint.s…"
↪   5 days ago     Up 5 days (healthy)     3306/tcp, 33060/tcp
↪                    mysql
c89de994abc6   oaisoftwarealliance/ims:latest          "asterisk -fp"
↪   5 days ago     Up 5 days (healthy)
↪                    ims
```

4.4. Part 4. Validate the Setup

In this section, you will validate the ATB setup using bi-directional UDP.

4.4.1. Step 1: Add the SIM User Profile

Modify the following files:

- `oai_db.sql`

  There are 3 UEs pre-configured in this file. To find them, search for `001010000000001` and add or edit them as needed.

- `./targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb-vnf.sa.band78.273prb.aerial.conf`

  Modify this file on the gNB server if you want to change the MCC and MNC in the gNB config file.

4.4.2. Step 2: Setup the UE and SIM Card

For reference, use the following: SIM cards – 4G and 5G reference software (open-cells.com)

Program the SIM Card with the Open Cells Project application "uicc-v2.6", which can be downloaded here.

Use the ADM code specific to the SIM card. If the wrong ADM is used 8 times, the SIM card will be permanently locked.

```
sudo ./program_uicc --adm 12345678 --imsi 001010000000001 --isdn 00000001 --
↪acc 0001 --key fec86ba6eb707ed08905757b1bb44b8f --opc
↪C42449363BBAD02B66D16BC975D77CC1 -spn "OpenAirInterface" --authenticate
Existing values in USIM
ICCID: 89860061100000000191
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 208920100001191
USIM MSISDN: 00000191
USIM Service Provider Name: OpenCells191
Setting new values
Reading UICC values after uploading new values
ICCID: 89860061100000000191
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 001010000000001
USIM MSISDN: 00000001
USIM Service Provider Name: OpenAirInterface
Succeeded to authentify with SQN: 64
set HSS SQN value as: 96
```

4.4.2.1 Commercial UE Configuration Setup

Install the "Magic IPERF" application on the UE:

1. To test with commercial UE, a test SIM card with **Milenage** support is required. The following must be provisioned on the SIM card and must match the Core Network settings: mcc, mnc, IMSI, Ki, OPc.
2. The APN on the commercial UE should be configured according to the Core Network settings.
3. Start the DNS. Core Network should assign a mobile IP address and DNS. If DNS is not assigned, set the DNS with the other Android app.

4.4.3. Step 3. Running End-to-End OTA

This section describes how to run end-to-end (E2E) traffic from the UE to the edge Core Network.

Note

The UE can connect as close as 2-3 meters, with a maximum range of 10-15 meters. The connection distance outside of buildings has not been verified.

4.4.3.1 CUE Connecting to 5G Network

Take the commercial UE out of airplane mode to start attaching the UE to the network. Make sure that the CUE is in airplane mode before starting OAI L2 stack.

4.4.3.1.1 Observe 5G Connect Status

Refer to the Preamble log in the cuphycontroller console output.

Check the Core Network log or commercial UE log to determine whether NAS authentication and PDU session succeeded.

4.4.3.2 Run E2E Iperf Traffic

Start `ping`, `iperf`, or other network app tests after the PDU session connects successfully.

You can install and run the “Magic IPerf” Android application on the commercial UE for this purpose.

4.4.3.2.1 Ping Test

Ping the UE from the CN:

```
docker exec -it oai-ext-dn ping 10.0.0.2
```

Ping from the UE to the CN:

```
ping -I 10.0.0.2 192.168.70.135
```

4.4.3.2.2 Iperf Downlink Test

Perform Iperf downlink test on the UE Side:

```
iperf3 -s
```

Perform Iperf downlink test on the CN5G Side:

```
# Test UDP DL
docker exec -it oai-ext-dn iperf3 -u -P 13 -b 80M -t 60 -c 10.0.0.2
#Test UDP bidirectional
docker exec -it oai-ext-dn iperf3 -u --bidir -P 13 -b 80M -t 60 -c 10.0.0.2
# Test TCP DL
docker exec -it oai-ext-dn iperf3 -P 13 -b 80M -t 60 -c 10.0.0.2
#Test TCP bidirectional
docker exec -it oai-ext-dn iperf3 --bidir -P 13 -b 80M -t 60 -c 10.0.0.2
```

4.4.3.2.3 Iperf Uplink Test

Perform Iperf uplink test on the UE Side:

```
iperf3 -s
```

Perform Iperf uplink test on the CN5G Side:

```
#UDP
docker exec -it oai-ext-dn iperf3 -u -R -b 130M -t 60 -c 10.0.0.2
#TCP
docker exec -it oai-ext-dn iperf3 -R -b 130M -t 60 -c 10.0.0.2
```

To stop the containers, use the following commands:

```
docker compose down
```

Note

ATB is a P5G cellular network; specific enterprise switching/routing/firewalls/policies might need integration support to enable access to the World Wide Web.

4.4.3.3 Monitor the CN5G Logs

```
docker logs oai-amf -f
```

You can also retrieve logs from all the CN functions with the below script.

```
cd ~/openairinterface5g/doc/tutorial_resources/oai-cn5g
./getLogs.sh
```

4.4.3.4 Capture PCAPs

```
docker exec -it oai-amf /bin/bash
tcpdump -i any -w /tmp/amf.pcap
```

Then copy the `.pcap` file out from the container.

```
docker cp oai-amf:/tmp/amf.pcap .
```

Chapter 5. Tutorials

The best way to get started with ATB is with the tutorials. The tutorials below will walk you through setup and some example use cases.

Topic	Title
Setup	Setting up 6G Research Testbeds (video; Dec 2024)
Deploy-ment	OpenAirInterface (OAI) 5G Core Network Deployment (video; Apr 2021)
	OpenAirInterface 5G Core Advance Deployment Using Docker-Compose (document; Jan 2023)

Chapter 6. Developer Zone

6.1. Developer Extensions

6.1.1. RIC Platform by Northeastern University

The Northeastern University (NEU) Wireless Institute of Things (WIoT) Institute is advancing the integration of O-RAN technology with NVIDIA's Aerial Testbed platform. One research topic is integrating an end-to-end (E2E) O-RAN E2 interface within the Aerial Testbed software stack. The integration leverages key components of the O-RAN ecosystem, including the O-RAN Software Community (OSC) RAN Intelligent Controller (RIC), and the OpenRAN Gym framework.

The integration enables two critical functionalities:

- **Streaming of key performance metrics (KPMs)**: The system can now transmit relevant performance data in real-time
- **Enforcement of control actions**: Decisions made by the xApps on the near-real time (Near-RT) RIC can be implemented swiftly.

Recent Developments

In July 2023, NEU showcased a significant milestone:

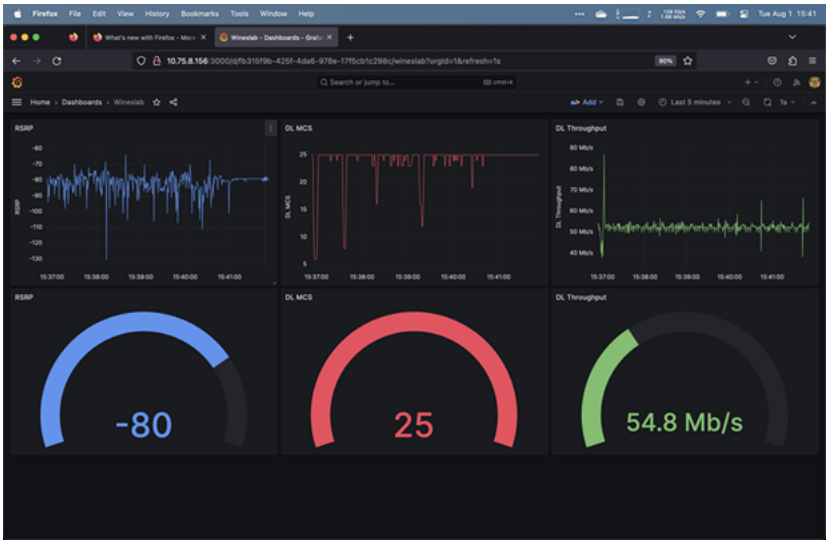
- A data-collection xApp running on an OSC RIC
- Deployed in a fully automated OpenShift cluster
- Connected to an InfluxDB database for telemetry storage
- Visualization of on a Grafana dashboard.

Ongoing Work

NEU is currently focused on enhancing the system's capabilities:

- **Near-RT Control**: The team is working to enable Near-RT control functionalities on the existing infrastructure
- **8-Node Deployment**: The institute is supporting an 8-node NVIDIA Aerial Testbed deployment, which serves as the testbed for these advancements.

This project represents a significant step forward in the implementation of O-RAN technology, potentially improving the flexibility, efficiency, and intelligence of radio access networks.



6.1.2. Kubernetes Service Management by Sterling SkyWave

Sterling SkyWave Service Management is a developer extension for NVIDIA ATB that enhances its capabilities with two key features:

- **Kubernetes (K8s) Service Orchestration**: Utilizes Helm for application management and includes two main Helm charts: skywave-service-management for gNB servers and oai-5g-basic for CN5G servers. The extension supports both single-node and multi-node deployment topologies, offering flexibility in network setup.
- **Service Monitoring**: Leverages open-source tools such as Grafana, Loki, Promtail, and Prometheus to provide comprehensive monitoring and visualization capabilities. It offers three default dashboards: ATB for gNB and UE status, GPU for NVIDIA Data Center GPU Manager (DCGM) metrics, and Host for system-level metrics.

The Sterling SkyWave Service Management extension is documented here.

6.1.3. Open5Gs by Northeastern University

Northeastern University has successfully integrated and validated Open5Gs, an advanced 5G open-source core network, in their experimental lab setup using an OpenShift cluster. This achievement represents a significant step forward in 5G network R&D.

Key Achievements

- **Microservice Architecture**: The core network is built on a microservice architecture, offering flexible deployment and scaling of individual network functions
- **Optimized User Plane Function (UPF)**: Delivers high-performance packet processing capabilities
- **User-Friendly SIM Management**: Offers easier management of user SIMs through a graphical interface
- **Network Slicing Support**: Enables the creating of multiple virtual networks on a single physical infrastructure
- **Deployment Flexibility**: Open5Gs demonstrates remarkable versatility in deployment options:

- Bare metal installation using standard Linux package managers
- Containerized deployment using Docker
- Virtualized approach utilizing Helm Charts on K8s and OpenShift

- **Performance and Compatibility**: When integrated with NVIDIA's Aerial Testbed platform, Open5Gs exhibited impressive performance:
  - **High Stability**: Maintained consistent operation during testing
  - **Sustained Performance**: Met the performance expectations set for the Aerial Testbed release
  - **MIMO Compatibility**: Successfully tested with OAI and 2-layer MIMO configurations
- **Implications for O-RAN Ecosystem**: This successful integration underscores the potential of the disaggregated O-RAN ecosystem. It demonstrates that components from different vendors can seamlessly integrate, fostering innovation and flexibility in 5G network deployments.

The Open5Gs implementation at Northeastern University showcases the power of open-source solutions in advancing 5G technology. By leveraging microservices architecture and supporting various deployment methods, Open5Gs provides researchers and developers with a robust platform for exploring next-generation mobile network capabilities.

6.1.4. n48 (CBRS) O-RU Interoperability by Rice University

The Rice University, Department of Electrical and Computer Engineering has made significant progress in enabling interoperability between NVIDIA ATB software with the Foxconn Citizens Broadband Radio Service (CBRS) O-RU (RPQN-4800E). This collaboration has yielded impressive results in lab testing, demonstrating the potential for advanced 5G and 6G research in the United States.

Key Achievements

- **Successful Testing**: The team achieved stable connectivity for over an hour in an indoor lab environment
- **Operational Spectrum**: Tests were conducted in a 100 MHz band (3.6-3.7 GHz)
- **Throughput Performance**: Achieved 250 Mbps DL and 50 Mbps UL speeds
- **Equipment Used**: Quectel RG520N UE module and OnePlus Nord 5G commercial handset

CBRS Spectrum Importance

The CBRS band (3.55-3.7 GHz) plays a crucial role in 5G deployment in the United States. The Federal Communications Commission (FCC) has opened this spectrum for shared access, implementing a three-tiered system:

- **Incumbent Users**: Government bodies
- **Priority Access License (PAL)**: Acquired through FCC auctions or secondary market sublicensing
- **General Authorized Access (GAA)**: Available when incumbent and PAL users are inactive

This shared access model, particularly the GAA tier, makes the CBRS band ideal for 5G research and development (R&D). It offers opportunities for experimentation without the high costs associated with PAL access.

6.1.5. GPU MIG Partition by Sterling SkyWave

The Sterling SkyWave GPU multi-instance GPU (MIG) Partition plugin is documented here.

Application Note

While running Aerial on a GPU partition device, the `mps_sm_*` parameters in the cuphycontroller config YAML file need to be adjusted accordingly such that the `mps_sm_*` value is not over the available streaming multiprocessors (SMs) of the selected MIG devices.

Please refer to the `mps_sm_*` configurations in `cuphycontroller_P5G_FXN.yaml` for the following cases:

- Running Aerial with MIG disabled

```
mps_sm_pusch: 108

mps_sm_pucch: 16

mps_sm_prach: 16

mps_sm_pdsch: 82

mps_sm_pdcch: 28

mps_sm_pbch: 18

mps_sm_srs: 16
```

- Running Aerial with MIG enabled on `mig-4g.48gb`

```
mps_sm_pusch: 42

mps_sm_pucch: 16

mps_sm_prach: 16

mps_sm_pdsch: 58

mps_sm_pdcch: 10

mps_sm_pbch: 8

mps_sm_srs: 8
```

- Running Aerial with MIG enabled on `mig-3g.48gb`

```
mps_sm_pusch: 40

mps_sm_pucch: 16

mps_sm_prach: 16

mps_sm_pdsch: 52

mps_sm_pdcch: 10
```

(continues on next page)

(continued from previous page)

```
mps_sm_pbch: 8

mps_sm_srs:  8
```

6.2. Featured Talks, Demos, and Sessions

6.2.1. Developer Radar Tech Talks

Date	Speaker	Description	Link
March 2024	Michele Polese, Anupa Kelkar	Learn about the developer journey of Northeastern University and Michele Polese, an early Aerial Testbed developer who went from an installed, configured, and operationalized 8-base station network to enabling multiple research streams, to then on-boarding and integrating a key network element, the RIC, as an Aerial Testbed extension and an opportunity to onboard ML-based applications for the developer community. Join Aerial Testbed developer and Assistant Professor Dr. Michele Polese from Northeastern University and NVIDIA Product Manager Anupa Kelkar as they share insights that showcase the potential of the platform capabilities, applications, and developer-extensions to jumpstart innovations in advancing wireless communications.	Northeastern Leads Open RAN Research
June 2024	Ravi P. Sinha Anupa Kelkar	Discover the O-RAN next Generation Research Group (nGRG) initiative aimed at advancing 6G R&D of future AI-native network technologies. Learn about nGRG, its roadmap, objectives, and the operational dynamics of its various research streams, which include 6G use cases, architecture, AI/ML, security, and a research platform for PoC projects. The talk also explores the evolution and life cycle management of a genuinely cognitive network within the O-RAN network ecosystem, along with integration of AI in the next-generation automated programmable framework enhanced by Large Language Models (LLMs).	O-RAN Alliance's Next Generation Research Group Framework for 6G
December 2024	CC Chong Joseph Boccuzzi	Learn how to set up a software-defined over-the-air wireless testbed for an end-to-end 6G research platform using Aerial Testbed.	Setting Up 6G Research Testbeds
March 2025	Davide Villa, Imran Khan, Florian Kaltenberger, Nicholas Hedberg, Rúben Soares da Silva, Stefano Maxenti, Leonardo Bonati, Anupa Kelkar, Chris Dick, Eduardo Baena, Tommaso Melodia, Michele Polese, Dimitrios Koutsonikolas	X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface A Detailed overview of multi-gNB network research testbed based on Aerial Testbed. Many performance metrics are provided, including GPU utilization and power metrics.	X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface

6.2.2. Developer Demos

Link	Description
Northeastern – X5G	X5G achieves a groundbreaking 8-node network deployment leveraging NVIDIA's ARC- OTA, integrating NVIDIA Aerial-CUDA Accelerated RAN for the PHY layer, accelerated on GPU. This innovative solution seamlessly combines with higher layers from the OAI open-source project through via the Small Cell Forum Functional Application Platform Interface (FAPI), setting a new standard for network efficiency and performance.
Fraunhofer – Aerial Spot Demo at Hannover Messe	Fraunhofer has successfully integrated an Open RAN network based on NVIDIA's ARC- OTA platform, showcasing its capabilities at the Y2024 Hannover Messe 6G-RIC booth in Germany. This integration demonstrates the potential of advanced wireless technologies and open-source solutions in real-world applications. ► **Key Components of the Demo** ► **NVIDIA Aerial Testbed Platform**: Utilized as the foundation for the Open RAN network ► **NVIDIA Aerial-CUDA Accelerated RAN**: Employed the library for the High PHY layer, leveraging GPU acceleration for enhanced performance ► **Open Air Alliance (OAI)**: Integrated for higher later functionalities, complementing the Aerial-CUDA Accelerated RAN.
AllBeSmart – AI-on-5G Demo	Allbesmart has implemented a cutting-edge demonstration showcasing the convergence of 5G and AI technologies using NVIDIA's Aerial Testbed platform. This innovative setup highlights the platform's capabilities for advanced wireless R&D. ► **Key Features of the Demo** ► **Integrated AI and 5G Processing**: A containerized AI video classification application runs concurrently with 5G PHY acceleration on a single NVIDIA GPU ► **NVIDIA Aerial Testbed**: Allbesmart operates a reference implementation of this platform, designed for 5G/6G R&D ► **Collaborative Effort**: The demonstration is the result of close collaboration between Allbesmart, the OAI team, and NVIDIA ► **Technical Highlights** ► **GPU Acceleration**: Utilizes NVIDIA's GPU technology to simultaneously handle complex 5G PHY protocols and AI workloads

6.2.3. Developer GTC Sessions

Link	Description
Advancing Connectivity: Democratizing 5G/6G Research With NVIDIA's Fully Open Programmable Network Stack	Today, we unveil a transformative solution poised to redefine the landscape of wireless communication. Our innovative platform is a beacon in the advancement of 5G+ and the forthcoming 6G networks, seamlessly blending digital and physical realities. This breakthrough goes beyond enhancing mobile broadband; it initiates an era of comprehensive digitalization, connecting humans, machines, and sensors like never before. As we enter an era where 6G introduces extraordinary intelligence, speed, and efficiency, our solution equips developers, researchers, and network providers with pivotal tools for this evolutionary leap. Join us to witness the unveiling of a technology that promises to reshape our wireless future, driving us toward a more connected, faster, and smarter world.
Programmable 5G and 6G networks	This panel will discuss the use cases and impacts of an AI/ML capable open and programmable next generation wireless network. Last GTC Aerial Testbed was launched as the first fully programmable 5G and 6G advanced wireless full stack. The full stack has enabled developers and researchers to experiment - simulate, prototype, and benchmark innovations with a hardware-in-the-loop OTA NR compliant platform enabled by NVIDIA accelerated compute. The panel will discuss product roadmap, virtualization and migration to cloud services, advanced developer use cases.
A Bridge to 6G - Aerial Research and Innovation Platform	NVIDIA and OAI experts provide an introduction to the first fully programmable Advanced 5G+ network as a sandbox – full-stack democratized platform for all researchers to simulate-prototype-benchmark optimizations, algorithms, and innovations rapidly in a deployed over-the-air NR standards compliant high performance operational network. This session will highlight platform vision, early adopter use cases, highlight C/C++ network programmability, provide OAI ISV gNB and CN overview and deep dive into specific ML examples that can jumpstart innovations.

6.3. Developer Use Cases

We love to see how Aerial Testbed is being used by developers, researchers, and the industry. Send an email to aerial-info@nvidia.com with your project description and links to the project and code repository (e.g. GitHub).

The following are example developer use cases.

6.3.1. ETH Zurich

Integrated Information Processing Group

The Integrated Information Processing (IIP) Group at ETH Zurich has successfully deployed a 5G vRAN system based on the NVIDIA Aerial Testbed platform. This system is fully software-defined and standards-compliant, enabling rapid prototyping and verification of novel baseband algorithms under real-world conditions.

Key Features and Advantages

- **Software-Defined System**: Allows implementation of novel baseband algorithms in CUDA for real-time execution and evaluation through OTA experiments.
- **Flexibility**: Offers the capability to extract real-time data from various parts of the signal processing chain, which is crucial for ML-assisted baseband algorithms.
  - The following UEs have been successfully tested in the system: iPhone 14 Pro, iPhone 15 Pro, iPhone 16E, Samsung Galaxy S23, Google Pixel 7, OnePlus Nord, Quectel RMU500EK
- **Cost-Effective**: Reduces development time and verification costs compared to hardware-based prototypes using FPGAs or ASICs.

Research Goals

- Develop novel ML-assisted baseband algorithms for future 5G and 6G wireless systems
- Optimize and validate solutions through OTA experiments on a real-world system
- Continue work on user positioning methods using self-supervised channel charting with channel state information (CSI)

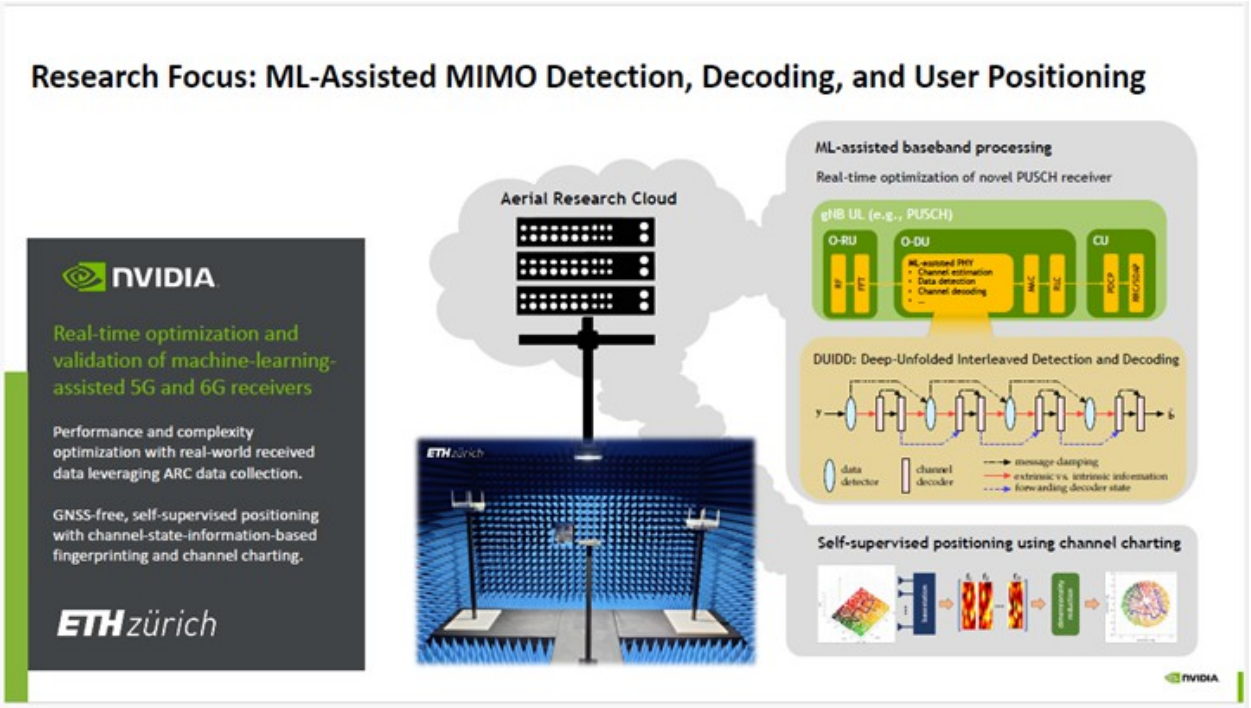
ML-Assisted Iterative MIMO Detection and Decoding

The group aims to implement their Deep-Unfolded Interleaved Detection and Decoding (DUIDD) receiver architecture on the NVIDIA platform. This approach:

- Fuses MIMO data detection and channel decoding with ML techniques
- Has shown 1.4 dB performance gains in simulations over classical iterative detection and decoding solutions
- Will be evaluated under realistic conditions to assess its efficacy and potential for adaptation to instantaneous system and channel conditions

This real-world 5G system provides a powerful platform for advancing wireless communications research beyond simulations, enabling the development and validation of innovative algorithms in realistic operational environments.

Real-World 5G System Blog



6.3.2. HHI Fraunhofer

6G-RIC Is Significantly Advancing Its Open Test Environment

Open source, E2E deployments are key, offering 6G-RIC researchers and associated startups a highly accessible and versatile platform for experimentation. This encourages innovation and facilitates the testing of emerging technologies, protocols, and applications. The integration of an Open RAN network, based on open-source technologies and NVIDIA ATB, marks a significant milestone for our project. The GPU-centric design is ideal for integrating AI/ML and expediting the creation of demonstrators, which once required significant development time.

6.3.3. Northeastern University

Northeastern University's Institute for the Wireless Internet of Things (WIoT) and its Open6G R&D Center have launched the first production-ready private 5G network fully automated through AI. This groundbreaking system is built on NVIDIA Aerial Testbed platform, enabling a fully virtualized, programmable O-RAN compliant network in a campus environment.

Key features of this innovative network include:

- Connectivity for 5G devices, supporting video conferencing, browsing, and streaming for experiential learning activities
- Built on open-source programmable components, utilizing compute solutions from partners like Dell Technologies and NVIDIA
- Employs zTouch, Northeastern's AI-based management, control, and orchestration framework for streamlined deployment and automated configuration
- Runs on Dell servers using OAI and Open5Gs for RAN and core network implementations
- Features base stations based on the NVIDIA Aerial Testbed, integrating a GPU-based PHY layer.

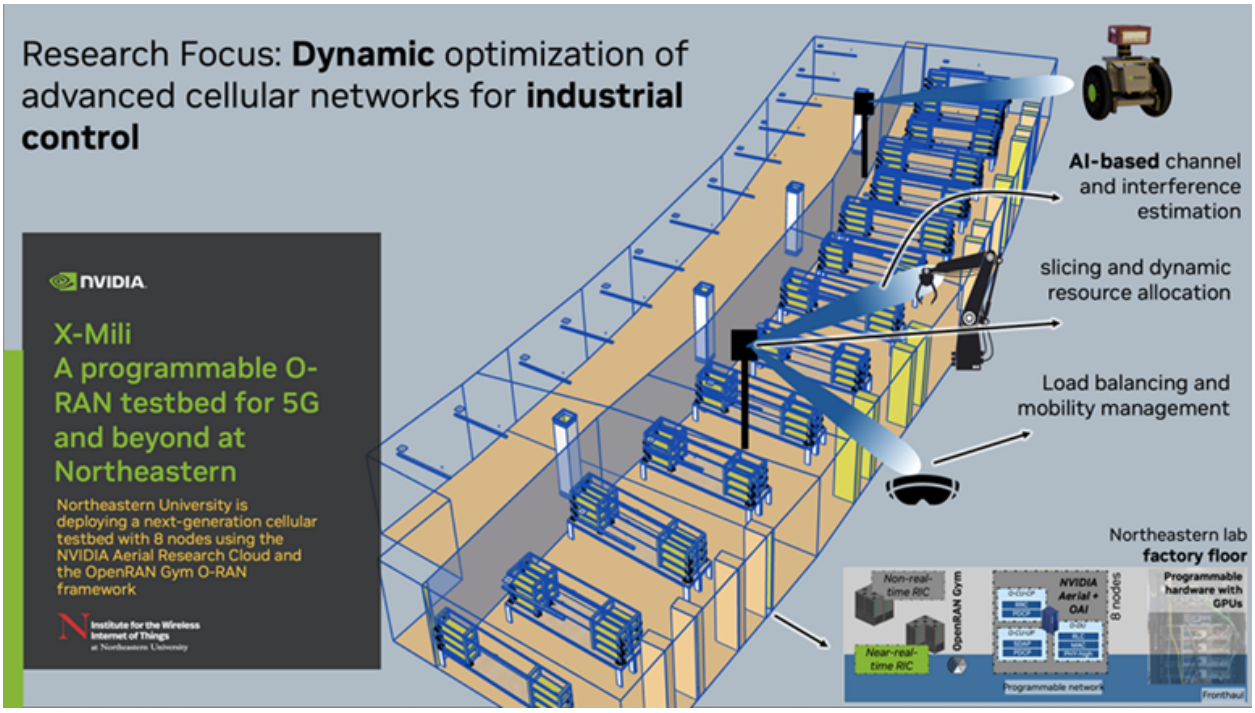
- The following UEs have been successfully tested in the system: OnePlus AC2003 Nord Samsung Galaxy S23, Sierra Wireless EM9191 NR 5G Modem, OAI Soft-UE.

The network showcases key features of next-generation wireless systems:

- Openness and programmability following the O-RAN architecture
- Resiliency and self-healing behavior through the zTouch automation framework
- Intelligent orchestration for managing xApps, rApps, and dApps.
- 55 UEs have been tested in a RF cabled test with the Keysight eLSU.

Currently deployed at Northeastern University's Boston campus, with plans to extend to the Burlington campus, this private 5G network offers unique opportunities for research in next-generation wireless technologies, including spectrum sharing mechanisms, AR/VR, E2E slicing solutions, and advanced security solutions.

There are more details for this project in this blog post. Visit https://wiot.northeastern.edu/ for information about the Northeastern Institute for the WIoT program.



6.3.4. OpenAirInterface Software Alliance

The OpenAirInterface (OAI) Alliance has demonstrated a 5G vRAN using NVIDIA Aerial CUDA-Accelerated RAN (formerly known as Aerial SDK) at the O-RAN virtual exhibition 2023. This demonstration showcases the integration of NVIDIA's L1 with OAI's L2+ to create an accelerated 5G vRAN.

Key Features of the Demonstrations

- **Hardware Setup**: The gNB (O-CU and O-DU) runs on a Dell server with an NVIDIA A100 Tensor Core GPU and ConnectX-6 DX SmartNIC
- **Network Configuration**: Uses O-RAN 7.2x fronthaul split, connecting to a commercial O-RU and a 5G phone
- **Containerized Environment**: Two containers run on the edge server - one for NVIDIA Aerial L1 and another for OAI L2+
- **Core Network**: Runs on a separate server with virtualized network functions (AMF, SMF, UPF) in different containers.

Technical Specifications

- Supports frequency range one, 30 kHz subcarrier spacing, 100 MHz bandwidth
- TDD config: 2.5ms periodicity, 3ms DL, 1ms UL
- Supports 2 layers of DL, 1 UL, and 1 cell.

Significance

This demonstration represents a shift towards software-defined, C/C++ programmable 5G base stations, enabling rapid prototyping and improved feature development without FPGA programming. It simplifies the development and testing of new 5G technology and applications, offering a cost-effective and performant alternative to traditional purpose-built custom hardware.

Learn more about this collaboration at the links below:

- Demonstration of NVIDIA Aerial SDK and OAI 5G vRAN and CN
- NVIDIA Aerial Testbed and OAI Demo Blog

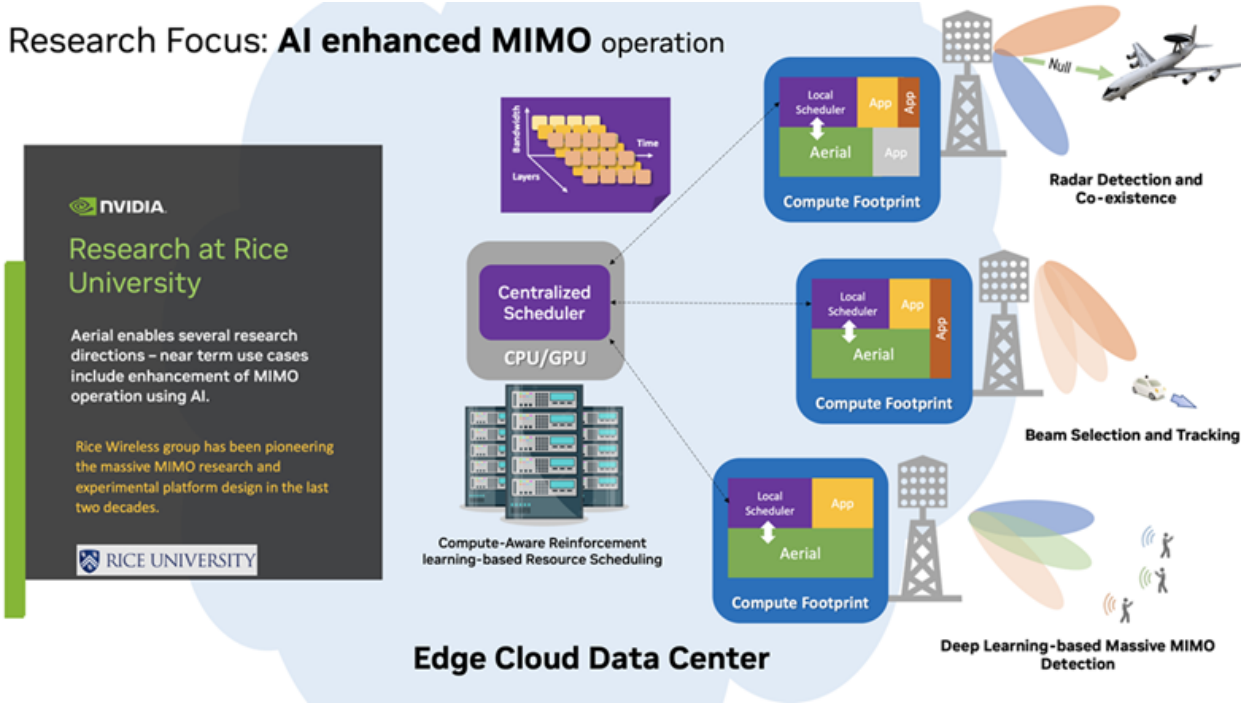
6.3.5. Rice University

Rice University outlines how NVIDIA Aerial Testbed platform makes several key contributions to the research described in this blog post:

- Aerial Testbed provides a 5G-compliant software-defined system that enables dataset generation at each layer of the network, which is crucial for training AI models
- The platform offers capabilities that help researchers pursue:
  - Representative datasets
  - E2E OTA performance benchmarking
  - Real-time implementation and performance evaluation of new algorithms
- For deep learning-based MIMO detection research, NVIDIA Aerial Testbed allows for:
  - Collection of real-world 5G-compliant data
  - Real-time implementation of AI-based detection algorithms on NVIDIA GPUs
- In radar detection and coexistence studies, the platform is used to:
  - Collect CSI from users affected by radar signals
  - Potentially implement real-time AI-based radar detection techniques
- For self-adapting vRANs research:
  - It enables benchmarking of wireless performance under varying compute loads
  - Allows investigation of GPU resource allocation for achieving specific data rates
  - Supports the development of AI-based schedulers that jointly allocate compute and radio resources

NVIDIA RC-OTA platform serves as a crucial tool for researchers to generate real-world data, implement and evaluate AI algorithms in real-time, and explore various aspects of 5G and beyond network optimization.

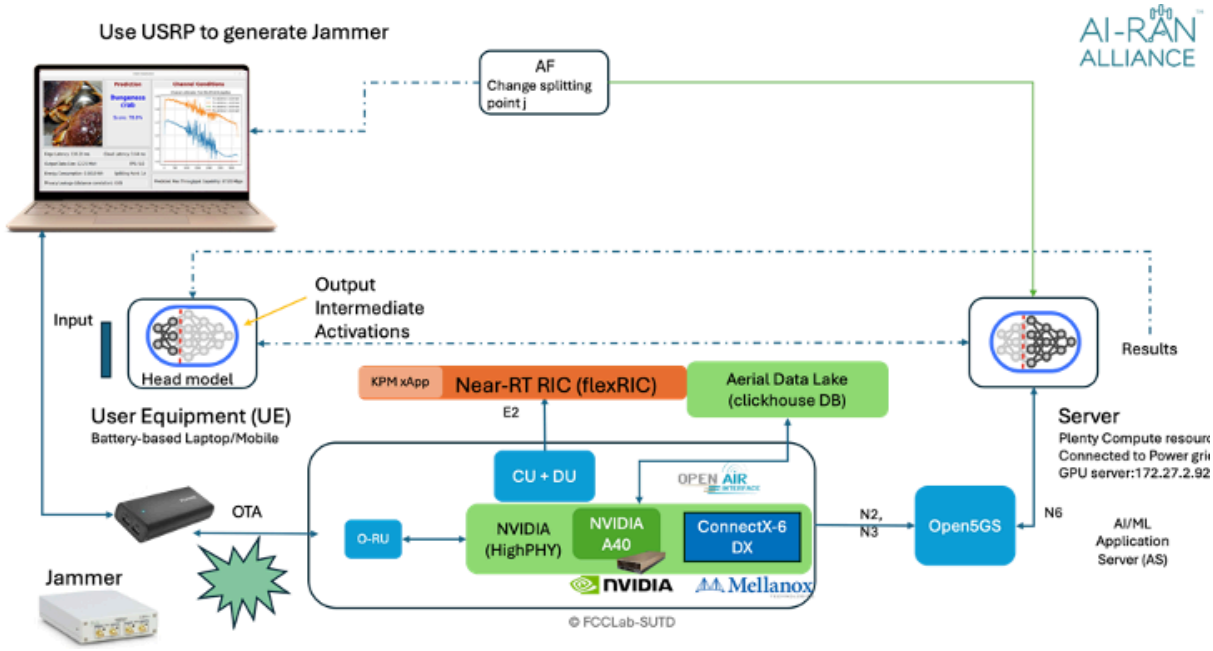
Visit https://wireless.rice.edu/ for information about the Rice Wireless program.



6.3.6. Singapore University of Technology and Design (SUTD) and Keysight Technologies

Under the umbrella of the AI-RAN Alliance, the Singapore University of Technology and Design (SUTD), in partnership with Keysight Technologies, used Aerial Testbed and Aerial Data Lake to implement a real-time OTA system that adaptively partitions an AI/ML image classification inference model between user equipment and infrastructure compute resources. The model split point is a function of the propagation channel which itself is determined by real-time spectrum sensing. This work shows how critical metrics such as privacy, end-to-end latency, energy efficiency and throughput can be optimized as a function of channel.

More information, including a video of the demonstration, can be found here.



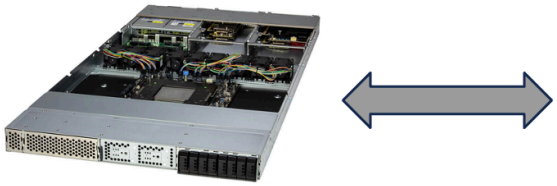
6.3.7. DeepSig Develops Algorithms for Learned Air Interface for 6G

This research work by AI-RAN alliance's member company, DeepSig is focused on developing and benchmarking an AI-native air interface for 6G physical layer. The goal is also to experiment with pilot-free or pilot-in-the-loop operations, jointly learning the modulation functions in the base station and in the UE. The approach aims to optimize the radio resource utilization for improved capacity over a wide range of specific and broad channel conditions.

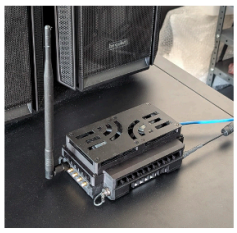
The hypothesis of this experiment is to challenge the current 5G air interface design that is model based with convenient assumptions on modulation, pilot, and frame design even though these are performance limiting. AI-native air interface allows AI to inherently design the waveform for a given site that will perform better. The approach allows AI to find the performance and capacity maxima by jointly learning and optimizing the waveform.

The setup uses the Aerial Testbed as the basis with a programmable UE implemented on a Jetson AGX Orin device:

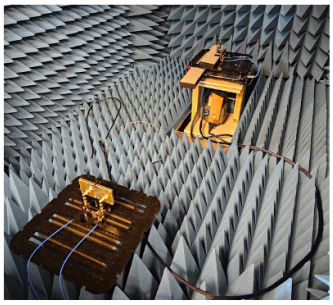
- Prototype 6G modem (OmniPHY) running on NVIDIA AI Aerial Platform which transceives information via neural encoding, receiving, and decoding
- Optimized to take advantage of GPU/DPU processing elements for low latency, high throughput, energy efficiency and for maximum capacity
- Can optionally perform continuous online optimization for conditions
- Tested Over the air using enterprise and mobile platforms



NVIDIA AI Aerial Platform (Base Station) /w OmniPHY Modem



Orin NX AI-Native PHY Mobile Device (Epiq G40)



DeepSig Lab Test Environment

The leaned air interface shows significant promise of improved site-specific performance as demonstrated by DeepSig at MWC2025:

More information on this can be found here.

6.4. Selected Developer News and Publications

Developer(s)/Author(s)	Title
O-RAN Spring 2024 Plugfest at Northeastern OTIC	Automating and Testing End-to-End O-RAN Systems
O-RAN Global Spring PlugFest 2024 at EURECOM	O-RAN Global Plugfest Spring 2024
Northeastern University	X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface (Journal extension under review - focus RIC)
Sterling	Sterling introduces SkyWave
Anupa Kelkar	A brief description of Aerial Testbed in two slides
NTIA, Office of Public Affairs	Northeastern and Rice university NTIA NOFO 1 win Biden-Harris Administration Award for Nearly $80M for Wireless Innovation
Eidgenössische Technische Hochschule Zürich	Real-World 5G System
Northeastern University	Northeastern University launches Fully Automated and Virtualized O-RAN Private 5G Network with AI Automation
TMCnet News	2023 Open Ran Product of the Year Award Winners
Davide Villa, Imran Khan, Florian Kaltenberger, Nicholas Hedberg, Ruben Soares da Silva, Anupa Kelkar, Chris Dick, Stefano Basagni, Josep M. Jornet, Tommaso Melodia, Michele Polese, Dimitrios Koutsonikolas	An Open, Programmable, Multi-vendor 5G O-RAN Testbed with NVIDIA Aerial Testbed and OpenAirInterface
Anupa Kelkar, Chris Dick	Introducing NVIDIA Aerial Research Cloud for Innovations in 5G and 6G
Florian Kaltenberger, Irfan Ghauri, Chris Dick, Anupa Kelkar, Lopamudra Kundu	Demonstration of NVIDIA Aerial SDK and OAI 5G vRAN and CN Virtual Exhibition
OpenAirInterface	OpenAirInterface Demonstrates 5G Virtual RAN with NVIDIA Aerial SDK
Jeffrey Andrews	Site Specific Deep Learning for the 6G Air Interface
Rahman Doost-Mohammady, Santiago Segarra, Ashutosh Sabharwal	Rice University Blog
Chris Dick	IEEE Keynote: The NVIDIA Roadmap tp AI-Infused 6G

Chapter 7. Resources

7.1. FAQs

Where is the hardware bill of material (BOM)?

The complete qualified ATB BOM is located in the *Procure the Hardware* chapter of the Installation Guide.

Does the platform support MU-MIMO?

ATB does not offer MU-MIMO integrated interoperability; however, the same platform is capable of adding software features for MU-MIMO.

Which frequency bands does the platform support?

ATB offers a tested reference for n48 and n78 sub-6 frequency bands.

Which RF parameters may require additional tuning for specific environments beyond the default config?

`pusch_TargetSNRx10`, `pucch_TargetSNRx10`, `ssPBCH_BlockPower`, `min_rxtxtime`, `TDD Patterns`, `RU Attenuations`

How can I apply for an experimental license in United States?

Review the application located at https://apps2.fcc.gov/ELSExperiments/pages/login.htm. If you have a program experimental license (https://apps.fcc.gov/oetcf/els/index.cfm), you can also use it for the Innovation Zone areas (Boston and the PAWR platforms) by submitting a request on that website.

Where can I find utility RF tools and calculators?

See Tools for RF Wireless.

Is GPS needed?

Yes, a GPS signal is necessary to drive precision timing for 5G networks. In case a GPS signal is inaccessible, the date command can be used as a workaround. This command is useful for those deployments where there is no timing reference (like GNSS) but needs Qg 2 to act as a GrandMaster (GM) to propagate time and synchronization over PTP to slave units.

When using two GMs, you can manually set the date and time on the first one and connect it's 1PPS/ToD output to the 1PPS/ToD port (configured as input) of the second GM. The second GM then outputs PTP messages with a clock class=6.

How can I check OS and kernel version and configuration?

Use the following commands:

```
cat /proc/cmdline
 BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-lowlatency root=UUID=d0bc583b-6922-4e70-
↪af14-92b624fe1bbe ro default_hugepagesz=1G hugepagesz=1G hugepages=16
↪tsc=reliable clocksource=tsc intel_idle.max_cstate=0 mce=ignore_ce
↪processor.max_cstate=0 intel_pstate=disable audit=0 idle=poll isolcpus=2-21
↪nohz_full=2-21 rcu_nocbs=2-21 rcu_nocb_poll nosoftlockup iommu=off intel_
↪iommu=off irqaffinity=0-1,22-23
uname -a
 Linux dev01 5.4.0-65-lowlatency #73-Ubuntu SMP PREEMPT Mon Jan 18 18:17:38
↪UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

7.2. Useful Shell Scripts

Use the following shell script to build cuBB (create the script in `/opt/nvidia/cuBB`):

```
#!/bin/bash
SCRIPT=$(readlink -f $0)
SCRIPT_DIR=$(dirname $SCRIPT)
echo running $SCRIPT
echo running $SCRIPT_DIR
export cuBB_SDK=${SCRIPT_DIR}
insModScript=${SCRIPT_DIR}/cuPHY-CP/external/gdrcopy/
echo $insModScript
cd $insModScript && make && ./insmod.sh && cd -
export LD_LIBRARY_PATH=$cuBB_SDK/gpu-dpdk/build/install/lib/x86_64-linux-gnu
echo $LD_LIBRARY_PATH | sudo tee /etc/ld.so.conf.d/aerial-dpdk.conf
sudo ldconfig
echo "perhaps you want to: "
echo "mkdir build && cd build && cmake .. "
mkdir -p build && cd $_ && cmake .. && time chrt -r 1 taskset -c 2-20 make -j
```

Use the following shell script to start cuphycontroller (create the script in `/opt/nvidia/cuBB`):

```
#!/bin/bash

sudo nvidia-smi -pm 1

sudo nvidia-smi -i 0 -lgc $(sudo nvidia-smi -i 0 --query-supported-
↪clocks=graphics --format=csv,noheader,nounits | sort -h | tail -n 1)

SCRIPT=$(readlink -f $0)
SCRIPT_DIR=$(dirname $SCRIPT)
echo running $SCRIPT
echo running $SCRIPT_DIR
export cuBB_SDK=${SCRIPT_DIR}
insModScript=${SCRIPT_DIR}/cuPHY-CP/external/gdrcopy/
echo $insModScript
cd $insModScript && ./insmod.sh && cd -
export LD_LIBRARY_PATH=$cuBB_SDK/gpu-dpdk/build/install/lib/x86_64-linux-gnu:
↪$cuBB_SDK/build/cuPHY-CP/cuphydriver/src
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/gdrcopy/src
```

(continues on next page)

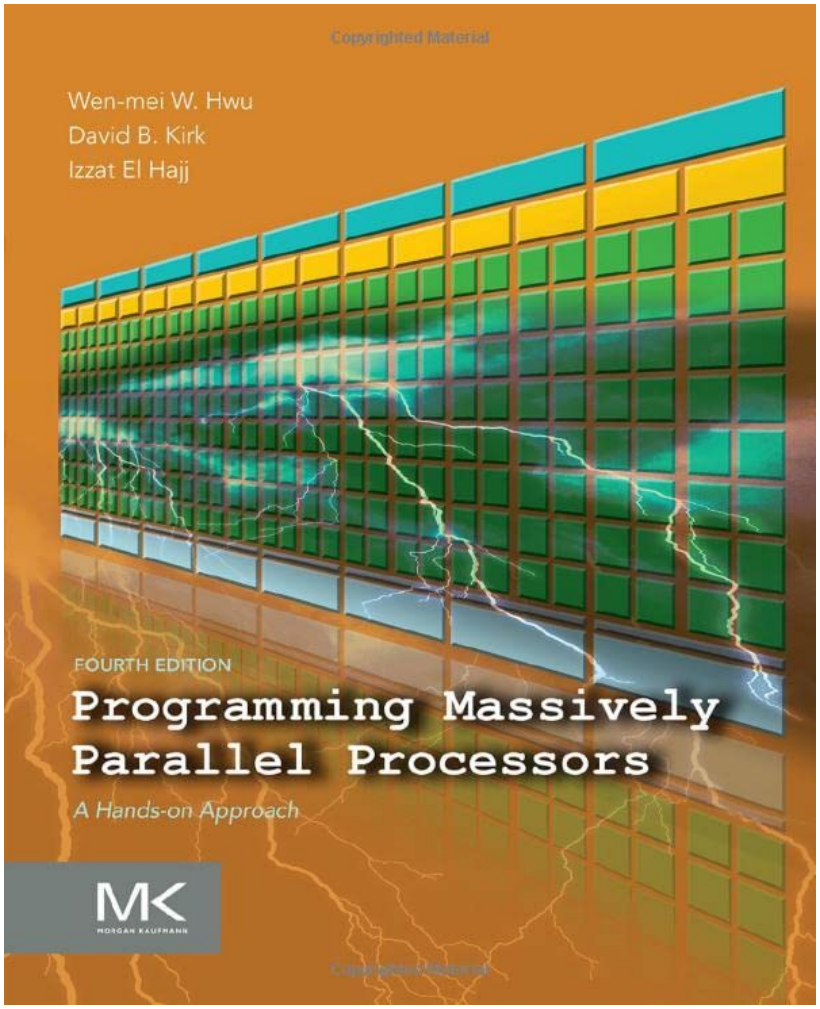
(continued from previous page)

```
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-
↪gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu
echo $LD_LIBRARY_PATH | sed 's/:/\n/g' | sudo tee /etc/ld.so.conf.d/aerial-
↪dpdk.conf
sudo ldconfig
export GDRCOPY_PATH_L=$cuBB_SDK/cuPHY-CP/external/gdrcopy/src
export CUDA_MPS_PIPE_DIRECTORY=/var
export CUDA_MPS_LOG_DIRECTORY=/var

sudo -E nvidia-cuda-mps-control -d
sudo -E echo start_server -uid 0 | sudo -E nvidia-cuda-mps-control
sleep 5
echo "perhaps you want to: "
echo "sudo -E LD_LIBRARY_PATH=$cuBB_SDK/gpu-dpdk/build/install/lib/x86_64-
↪linux-gnu/ ./build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf
↪P5G_SCF_FXN"
sudo -E ./build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf P5G_SCF_
↪FXN
sudo ./build/cuPHY-CP/gt_common_libs/nvIPC/tests/pcap/pcap_collect
```

7.3. Recommended Reading Material

Programming Massively Parallel Processors by David B. Kirk and Wen-mei W. Hwu



7.4. Hands-on CUDA-C

To learn CUDA and learn teach it to others, refer to the NVIDIA/UIUC Accelerated Computing Teaching Kit, which outlines each module's organization.

7.5. Additional Help

- Join the NVIDIA 6G Developer Program
- Visit the GitHub Discussions forum
- Contact us at aerial-info@nvidia.com

Chapter 8. Licensing

8.1. Aerial CUDA-Accelerated RAN

Refer to the Aerial CUDA-Accelerated RAN Acknowledgements page for license and copyright information.

8.2. OAI License Model

Aerial Testbed employs the OpenAirInterface L2 from the 5G RAN Project Group and the OAI Core Network (CN) from the OAI 5G Core Network Group. These software components are obtained via the OAI open source repository. Refer to the OAI license model for details.

Copyright