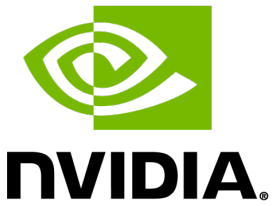


Aerial RAN CoLab Over-the-Air

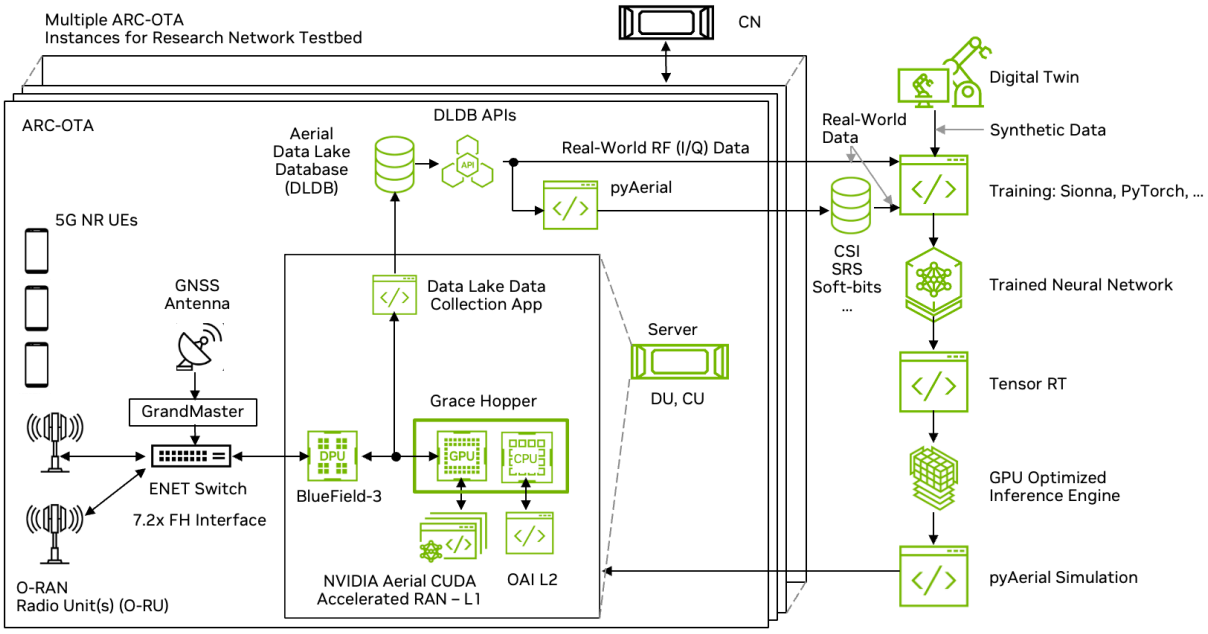
Release 1.5.1

NVIDIA Corporation

Mar 19, 2025

Contents

Chapter 1. Introduction



Meet NVIDIA Aerial RAN CoLab Over-the-Air (ARC-OTA): The Ultimate AI-Enhanced Wireless Development Platform

NVIDIA ARC-OTA is a groundbreaking platform that revolutionizes advanced wireless development, now featuring powerful AI-RAN capabilities. This versatile sandbox environment offers developers unprecedented access and flexibility for experimentation and innovation in wireless technologies.

Key Features

- **Complete Source Code Access**: Developers can dive deep into the platform's inner workings, enabling thorough customization and experimentation
- **Rapid Validation**: Quick turnaround for testing and benchmarking results, accelerating the development process
- **O-RAN Compliance**: Built on principles of virtualization, disaggregation, and full software programmability, ensuring compatibility with Open RAN standards
- **AI-RAN Integration**: Leveraging NVIDIA's AI Aerial hardware and software solution for enhanced capabilities

Advanced Capabilities

- **GPU-Accelerated PHY**: Full inline acceleration of all PHY functions in addition to AI/ML integration in the RAN
- **Commercial-Grade Foundation**: Built upon NVIDIA Aerial CUDA-Accelerated RAN, providing a robust and reliable base for development
- **Cloud-Native Architecture**: Designed for seamless integration with modern cloud infrastructure, supporting 5G and 6G network stacks.

Developer Benefits

- **Unified Workload Handling**: ARC-OTA efficiently manages both RAN and AI workloads, streamlining the development process
- **Continuous Innovation**: The platform's roadmap includes NVIDIA-qualified tools, blueprints, and emerging developer extensions to foster community-driven innovation
- **Empowering Creativity**: With its AI-RAN capabilities, ARC-OTA enables developers to push the boundaries of advanced wireless technologies, encouraging exploration and breakthrough innovations.

By combining cutting-edge AI capabilities with a flexible, open development environment, the ARC-OTA platform stands at the forefront of wireless technology innovation, empowering developers to shape the future of 5G, 6G, and beyond.

1.1. Latest Product Updates

Product Area	Updates
Software Release Manifest	ARC-OTA 1.5.1 is based on Aerial CUDA-Accelerated RAN 24-1 + OAI 2024.w21 tag
Platform	This release supports the following platforms: 1. Supermicro GH200 + BF3 2. Gigabyte (A100 + CX6-DX) 3. Dell R750 + A100X
New	**First** release for ARC-OTA on SMC-GH platform
DL Peak Performance Improvements	**First** release for ARC-OTA with 1 Gbps (DL) peak performance
Key Release KPIs	4 layers DL –> peak performance improvement (previously 460 Mbps)
	SMC-GH DL 1.03 Gbps / UL 130 Mbps, 4DL/1UL + CBRS RU soak testing 10+ hours
	Dell R750 / Gigabyte DL 900 Mbps / UL 110Mbps 4DL/1UL (4 hours)
SDK Manager Network-as-a-service tooling	SDK Manager support for SMC-GH server
Other features	Multi-UE qualified OTA (6 UEs)
	Sub-6 n48 CBRS O-RU integrated
	Multi-UE CSI dataset recipe (w/ pyAerial and Aerial Data Lake integrated)

1.2. Software Release Manifest

The following table outlines the software versions used for ARC-OTA 1.5.1.

Component		Version
Aerial CUDA-Accelerated RAN	Layer 1	24-1
	Data Lakes	24-1
	pyAerial	24-1
OAI[1]	gNB	2024.w21
	CN[2]	2024.w21
NVIDIA SDK Manager		v2.1
Foxconn O-RU n78 and CBRS		v3.1.15q.551v0706-oam

[1] For additional context, developers can also review the artifacts in the 2024.w21+ARC1.5 branch.
[2] The Grace Hopper platform requires OAI CN version 2024-June.

Chapter 2. Release Notes

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

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

2.1.1. 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`.

2.1.2. 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)

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

2.2.1. 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`).

2.2.2. Performance Improvements

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

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

2.3.1. 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

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

2.4.1. 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.

2.5. 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

ARC-OTA is an AI/ML enabled E2E real-time OTA network testbed for AI-RAN research. As shown in the figure below, ARC-OTA 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 CN can run on the same server that hosting L1 and L2, or it can be supported on a separate x86 or ARM server. An NVIDIA NIC, in this case the BF3 data processing unit (DPU), connects to a fronthaul switch using a 7.2x interface. The switch connects to one or more O-RUs for single or multi-cell operation respectively.

All CUDA source code is available for L1 and C code for L2 and CN. With access to source for all components of the software stack, 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.

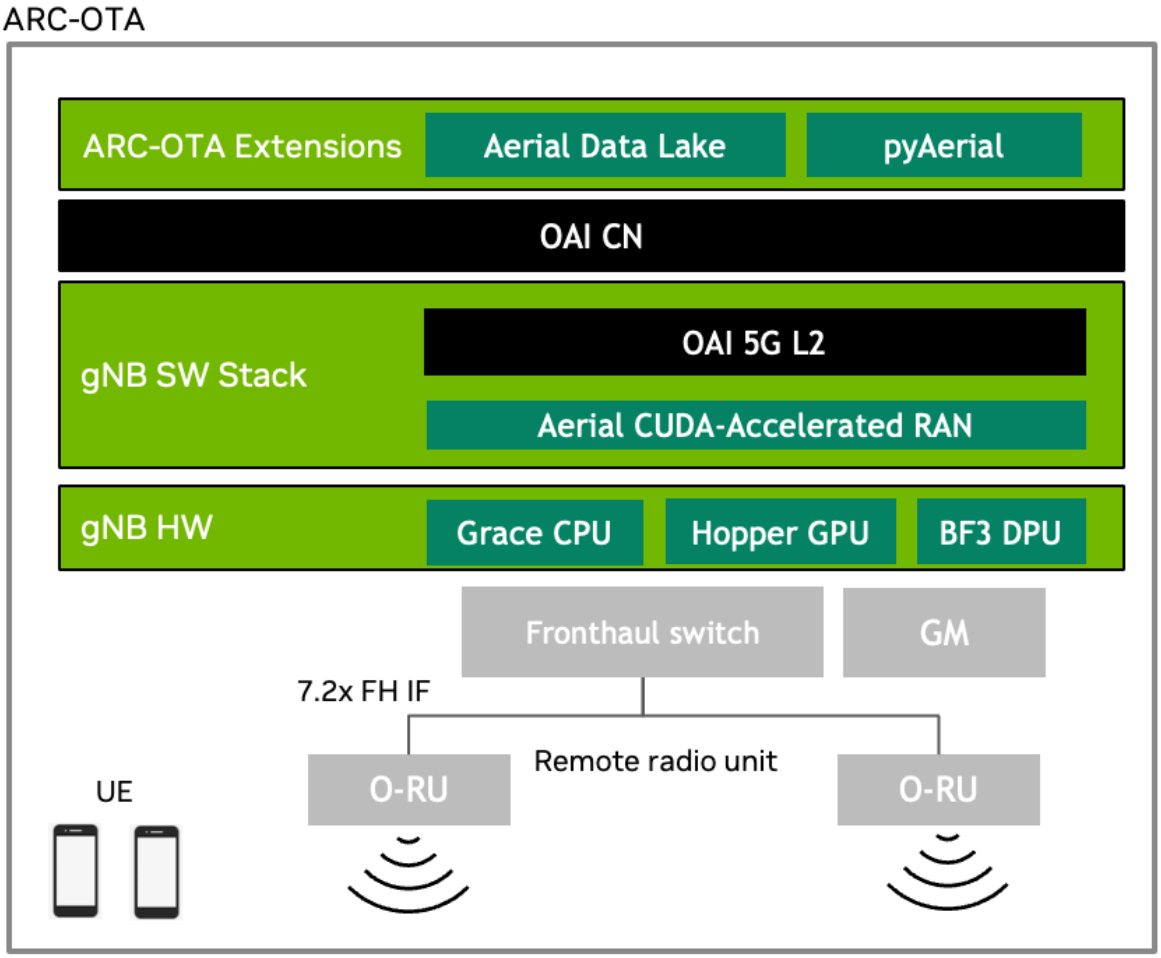
With an ARC-OTA testbed you can test ML algorithms at all layers of the stack. You can bring ML to the physical layer, to layer 2, and benchmark them in a live network. As real O-RUs are used in the testbed, your algorithms can be verified and benchmarked in the context of real-world wireless channels, 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. ARC-OTA 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.

ARC-OTA is built with an eye to enabling AI/ML research. In particular, it supports the capture of OTA data for use in training pipelines. Data collection is facilitated using the NVIDIA Aerial Data Lake app, which is part of NVIDIA Aerial AI Radio Frameworks. Aerial Data Lake runs as an app on the Distributed Unit (DU). It collects the uplink I/Q samples from the O-RU(s) that are delivered to the cuPHY baseband over the 7.2x fronthaul interface and writes them to a database. FAPI meta information exchanged between L1 and L2 are also collected and populated in the database and can be used for indexing into and extracting data from the Data Lake database.

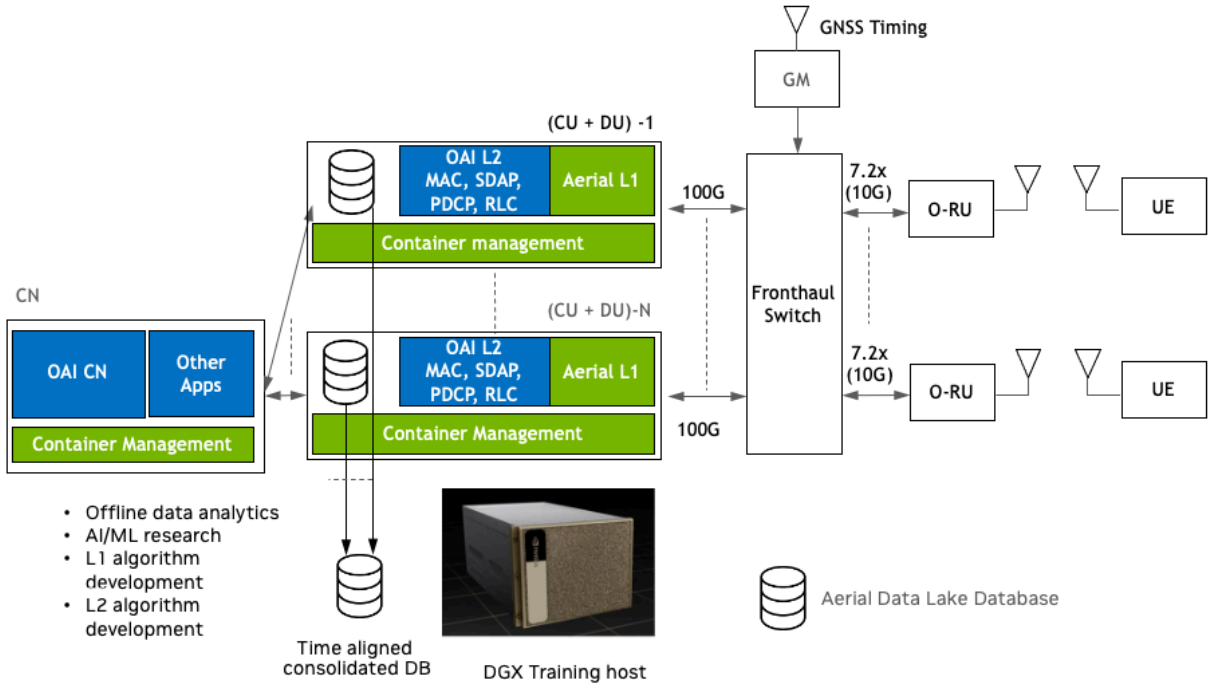
While the uplink I/Q samples, together with L2 meta information, are useful for some types of algorithm development, each type of ML, or for that matter non-ML, algorithm design requires a data set tailored to the use-case at hand. This is where pyAerial helps. NVIDIA pyAerial is another tool within NVIDIA Aerial AI Radio Frameworks. 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 the 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 ARC-OTA, including add-on services such as Aerial Data Lake and pyAerial.



As shown in the figure below, multiple ARC-OTA instances can be used to assemble a multi-cell research network. Each network node can collect data using Aerial Data Lake. The distributed Data Lake databases can be time-aligned and consolidated into a multi-cell training database using the time stamps associated with database records.



3.1. Key Features and Specifications

The configuration and capabilities of ARC-OTA 1.5.1 are outlined in the following sections.

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 55	
Number of UEs/TTI	2	
Frame structure and slot format	DDDDDDSUUU	
	DDDSU	
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	SMC-GH	DL: ~1.03Gbps
		UL: ~125Mbps
	Dell R750 / Gigabyte | DL: ~800Mbps +———— | UL: ~110Mbps	
Bi-directional UDP Traffic	> 10+ hours exercised (SMC-GH) > 4.0 hours exercised (Dell R750 + A100X) > 4.0 hours exercised (Gigabyte + A100 + CX6-DX)	

Tip

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

3.2. Product Features

Feature	Description
Full stack software	A 3GPP Release 15 compliant and O-RAN 7.2 split 5G SA 4T4R wireless stack, with all network elements from RAN and 5G Core. Aerial CUDA-Accelerated RAN is integrated with the OAI Distributed Unit (DU), Centralized Unit (CU), or a 5G NR gNB and 5G Core Node(CN) network elements.
Radio network hardware components	The commercial off-the-shelf (COTS) hardware BOM used for NVIDIA qualification is available in the *OTA-qualified HW BOM manifest*.
Source code access	Complete access to source code in C/C++, from Layer 1 through 5G core to jump start customizations and next-generation algorithm research. Review the *Licensing* section for more details.
Developer contributions	To accelerate innovation, developer extensions and contributions are welcome. Refer to the *Developer Zone* section for more details.
AI frameworks	AI frameworks and tools are integrated to ease the AI/ML developer journey for advanced wireless research. For example, pyAerial and Aerial Data Lakes has been integrated with ARC-OTA.

3.2.1. 5G NR gNB Features

Component	Capabilities
gNB PHY	Aerial cuPHY Layer 1 adheres to 3GPP Release 15 standard specifications to deliver the following capabilities. PHY capabilities include the following: ▶ Error detection on the transport channel and indication to higher layers ▶ Forward error correction (FEC) encoding/decoding of the transport channel ▶ Hybrid automatic repeat request (ARQ) soft combining ▶ Rate matching of the coded transport channel to physical channels ▶ Mapping of the coded transport channel onto physical channels ▶ Power weighting of physical channels ▶ Modulation and demodulation of physical channels including ▶ Frequency and time synchronization ▶ Radio characteristics measurements and indication to higher layers ▶ Multiple Input Multiple Output (MIMO) antenna processing ▶ Transmit (TX) Diversity ▶ Digital and Analog Beamforming ▶ Radio frequency (RF) processing 3GPP standards specifications that define the Layer 1 compliance are: ▶ TS 38.211 (38.211 v15.8.0) numerologies, physical resources, modulation, sequence, signal generation ▶ TS 38.212 (38.212 v15.8.0) Multiplexing and channel coding ▶ TS 38.213 (38.213v15.8.0) Physical layer procedures for control ▶ TS 38.214 (38.214v15.8.0) Physical layer procedures for data ▶ TS 38.215 (38.215v15.8.0) Physical layer measurements ▶ TS 38.104 (base station radio Tx and Rx) Base Station (BS) radio transmission and reception Aerial CUDA-Accelerated RAN complies with O-RAN fronthaul (FH) control, user, and synchronization (CUS) specification version 3 (version 4 for power scaling). Aerial CUDA-Accelerated RAN complies with northbound interfaces adopted by the industry based on Small Cells Forum for Layer 1 and Layer 2 (SCF FAPI).
gNB MAC	▶ MAC -> PHY configuration using NR FAPI P5 interface ▶ MAC <-> PHY data interface using FAPI P7 interface for BCH PDU, DCI PDU, PDSCH

3.2.2. 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)

3.2.3. 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

3.3. Product Blueprints

To ease developer onboarding, this section provides reference blueprints with key ingredients that for creating a tested product prototype: a full-stack development testbed designed to accelerate innovation in wireless networks and provide new insights on experiments and research. The NVIDIA ARC-OTA, enhanced with AI-RAN capabilities, is a versatile platform available to all advanced wireless developers.

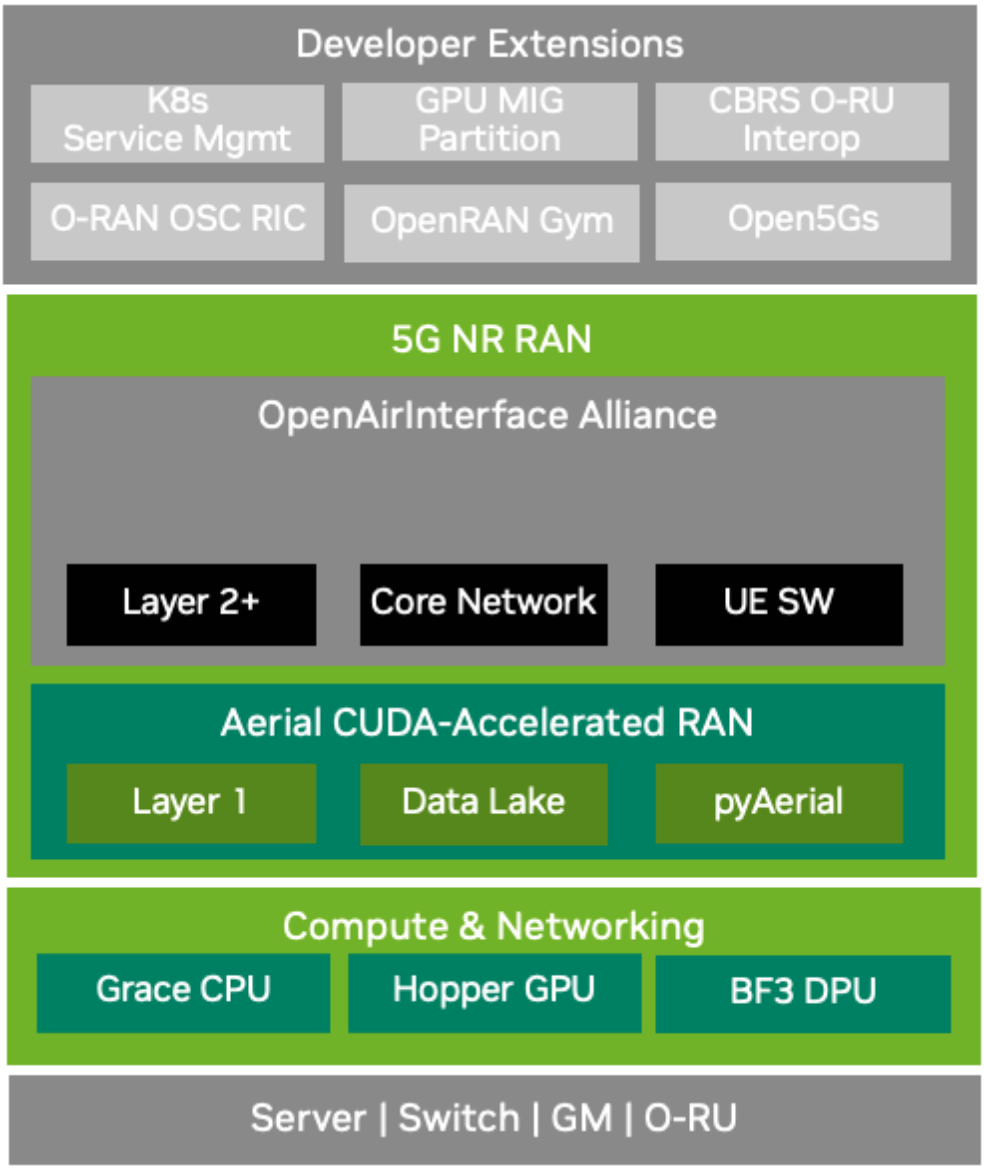
ARC-OTA with AI-RAN integration offers several advantages:

- **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 functionalies
- **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**: ARC-OTA 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, ARC-OTA 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.

3.3.1. 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	Aerial 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

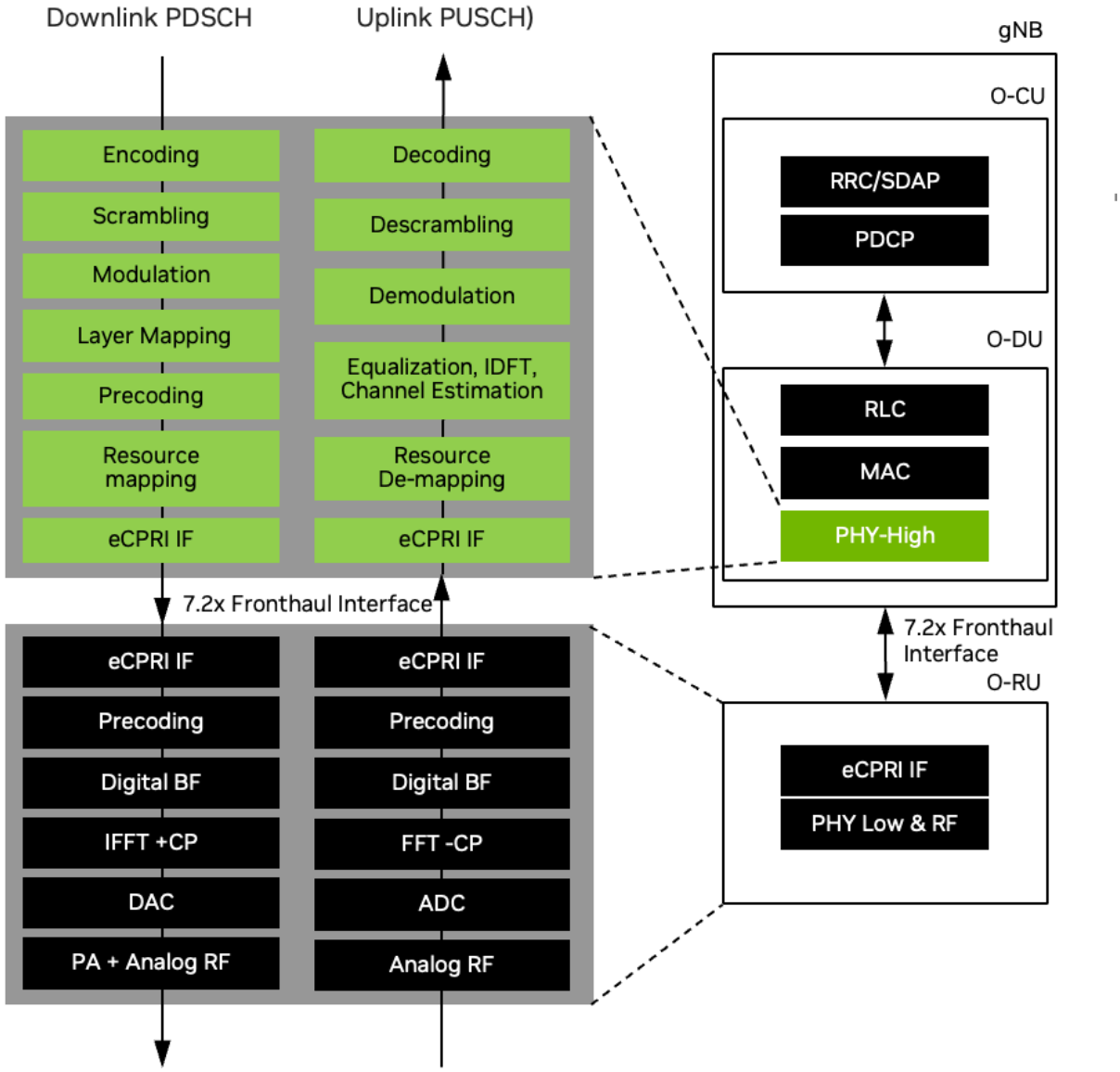
Note

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

3.3.2. ARC-OTA 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.

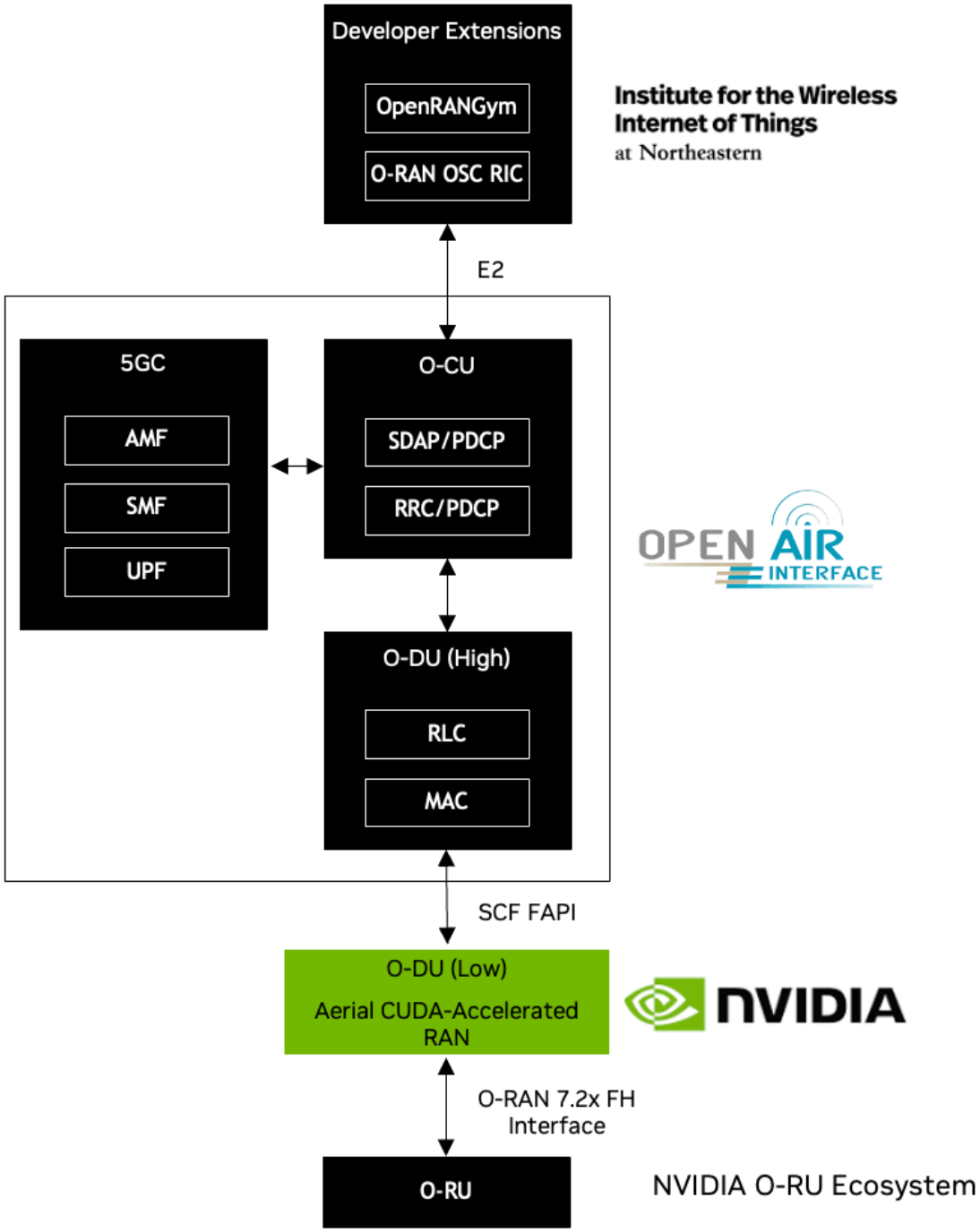
O-RAN's split-RAN concept disaggregates the RAN into multiple functional components: O-RAN Central Unit (O-CU), O-RAN Distributed Unit (O-DUC) and O-RAN Radio Unit (O-RU), as shown in the figure below. These components can be deployed on different hardware and software platforms and can be interconnected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



ARC-OTA 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-RU's supplied from the NVIDIA O-RU ecosystem partners, Layer-1 is NVIDIA Aerial-CUDA Accelerated RAN, the 5G Core, O-CU and O-DU (high) are 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	Features
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
Foxconn	O-RU Low PHY
Others	Handsets (Apple iPhone 14, Samsung S23), Viavi Qualsar Grandmaster, Dell FH switch, Supermicro GH200 server.

ARC-OTA leverages the O-RAN 7.2x split, which divides the protocol stack into the following:

Component	Description
O-RU	The Open RAN radio-unit (O-RU) is responsible for the physical layer-low processing, in addition to digital IF and RF signal processing and analog-to-digital conversion and digital-to analog conversion
O-DU	The Open RAN distributed-unit (O-DU) is responsible for the physical layer-high proessing the higher-layer processing, including MAC, RLC, and PDCP
Fronthaul interface	The O-RAN Alliance specifies the fronthaul interface between the O-DU and O-RU based on the 7.2x split. This interface supports control, user, synchronization (CUS), and management (M) planes

3.3.3. Data Collection on ARC-OTA with Aerial Data Lake

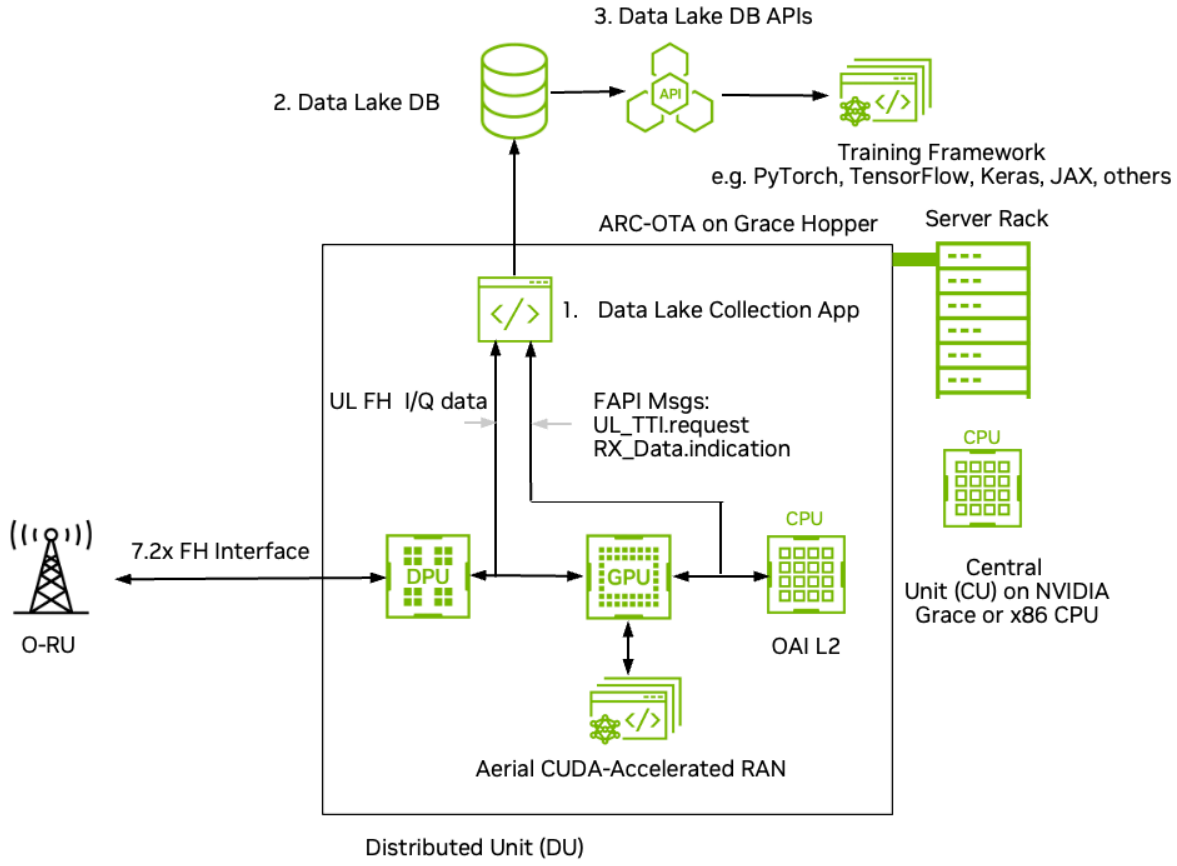
6G will be AI-native. AI/ML will be deeply embedded across all layers of next-generation networks—from radio and baseband processing to the network core, encompassing system management, orchestration, and dynamic optimization. Realizing this vision of a software-defined, AI-native wireless communication infrastructure will require GPU hardware alongside advanced programming frameworks.

One of the most exciting areas of research is the application of AI/ML in the physical layer. Data is the fuel of AI. While synthetic data generation, powered by tools like Aerial NVIDIA Aerial Omniverse Digital Twin (AODT) and NVIDIA Sionna/Sionna RT, play a crucial role in AI-driven research, real-world OTA waveform data from live systems is equally essential. This is where Aerial Data Lake (ADL) comes in—a cutting-edge data capture platform designed to collect OTA RF data from the NVIDIA ARC-OTA testbed, enabling deeper insights and advancements in AI-native wireless 6G.

ARC-OTA can collect, in real-time, OTA data using the ADL data collection application shown in the figure below. ADL consists of 3 parts:

1. The data collection app (DCA) running on the Grace CPU
2. The Aerial Data Lake Database (DLDB)
3. A set of DLDB APIs used for retrieving data from the DLDB.

The DCA captures UL I/Q samples from 7.2x fronthaul interface in addition to a sub-set of the FAPI messages exchanged between L2 and L1.



Aerial Data Lake has the following key features:

- Real-time capture of RF data from OTA testbed
  - ADL is designed to operate with the NVIDIA ARC-OTA network testbed. I/Q samples from O-RUs connected to the GPU platform via a 7.2x split fronthaul interface are delivered to the host CPU and exported to the DLDB.
- Powerful database access APIs
  - The layer-2 messages Rx_Data.indication and UL.TTI.request are filtered from the layer-2/layer-1 FAPI message stream and exported to the database. The fields in these data structures form the basis of the database access APIs to extract data from the database for training in ML frameworks such as PyTorch, Sionna and JAX to name a few.
- Scalable and time coherent over arbitrary number of base stations
  - The data collection app runs on the same CPU that supports the DU. It only consumes a small number of CPU cores. Because each base station is responsible for collecting its own uplink data the collection process scales as more base stations are added to the ARC-OTA network testbed – refer to the figure above in the Product Description section. Databases are time-stamped and so data collected on multiple base stations can be used in a training flow in a time-coherent manner.
- Used in conjunction with pyAerial to generate training data for neural network physical layer designs

- ADL is designed to be used in conjunction with NVIDIA pyAerial CUDA accelerated Python library. Using the DLDB APIs, pyAerial can access RF samples in a DLDB and transform those samples into training data for all of the signal processing functions in an uplink or downlink pipeline.

Information about installing and using Aerial Data Lake and pyAerial can be found at the NVIDIA AI Aerial Site.

Chapter 4. Installation Guide

The following sections will guide you through the process of integrating and deploying ARC-OTA:

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 – Configure gNB Server	Install the kernel command line specific to ARC-OTA.
Part 4 – Install ARC-OTA Using NVIDIA SDK Manager	Use NVIDIA SDK Manager to install ARC-OTA.
Part 5 – Validate the Setup	Validate the setup using bi-directional UDP.
App Note – ARC-OTA Step-by-Step Debug Command Line	This installation method is for debug purposes only. Instead of using SDK Manager to install ARC-OTA, all necessary software components are installed manually.

4.1. Part 1. Procure the Hardware

Procure all the hardware listed in the following BOM.

4.1.1. ORAN 7.2x Reference Hardware Components

ARC-OTA covers a wide range of use cases for developers in advanced wireless. Upon integration and qualification, these ORAN 7.2x split-aligned BOM hardware components are shared via the reference below.

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		
ARC-OTA gNB	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)
	Dell PowerEdge R750 Server + A100X	GPU	A100X
	Gigabyte Edge E251-U70 Server	CPU	Intel Xeon Gold 6240R, 2.4GHz, 24C48T
		GPU	A100
		Memory	96GB DDR4
		Network	MLX CX6-DX MCX623106AE-CDAT
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 (formerly Qulsar)		
O-RUs supported	**ORU**	**Freq Band**	
	Foxconn RPQN-7801E (4T4R)	(indoors) 3.7GHz - 3.8GHz	
	Foxconn RPQN-4800E (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		

continues on next page

Table 1 – continued from previous page

5G Infra Component	Hardware Manifest
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
Transceivers	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 ARC-OTA)

4.2. Part 2. Configure the Network Hardware

Note

Refer to the NVIDIA SDK Manager resources for setup and installation of ARC-OTA.

The network hardware is configured in the following steps.

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

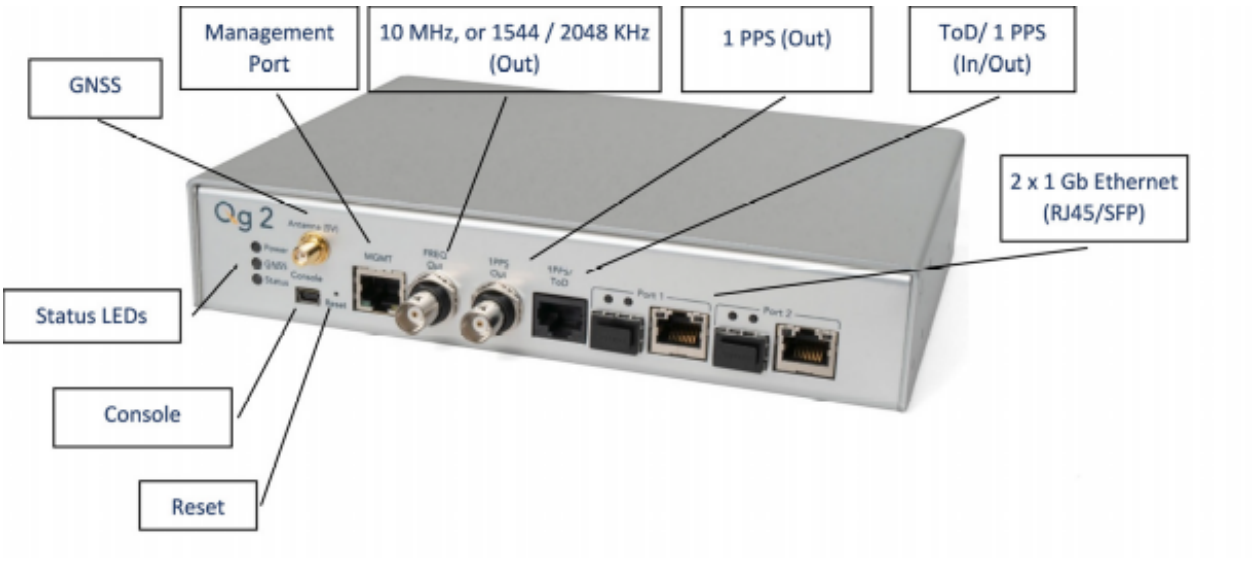
4.2.1. Part 2.1 - Setup the VIAVI Solutions GrandMaster

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 ARC-OTA system.

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

Front Panel

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



Back Panel

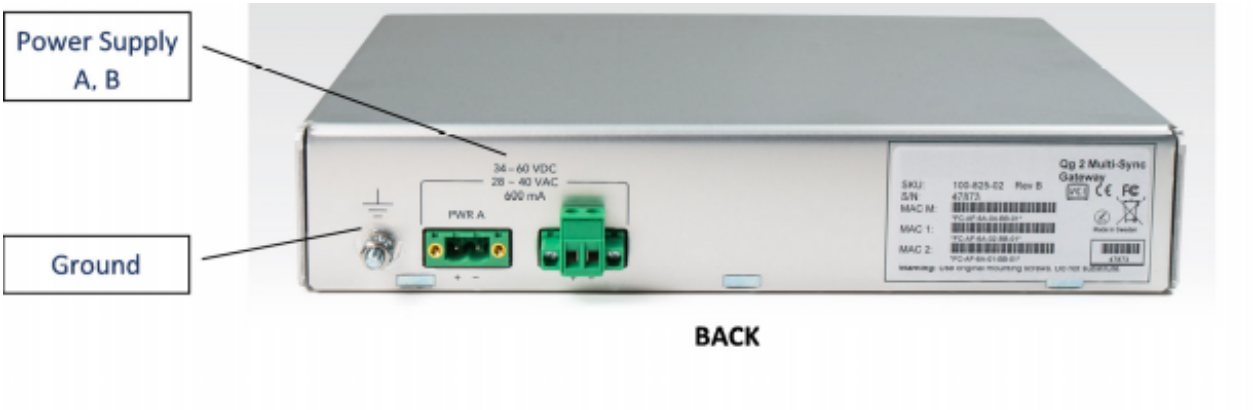


Fig. 1: Figure credit: VIAVI QG2 Multi-Sync Gateway Users Guide

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)#
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
```

(continues on next page)

(continued from previous page)

```
↪[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

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

7. Configure the RU, gNB, CN, and MEC 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
```

(continues on next page)

(continued from previous page)

```
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
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 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 this documentation 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

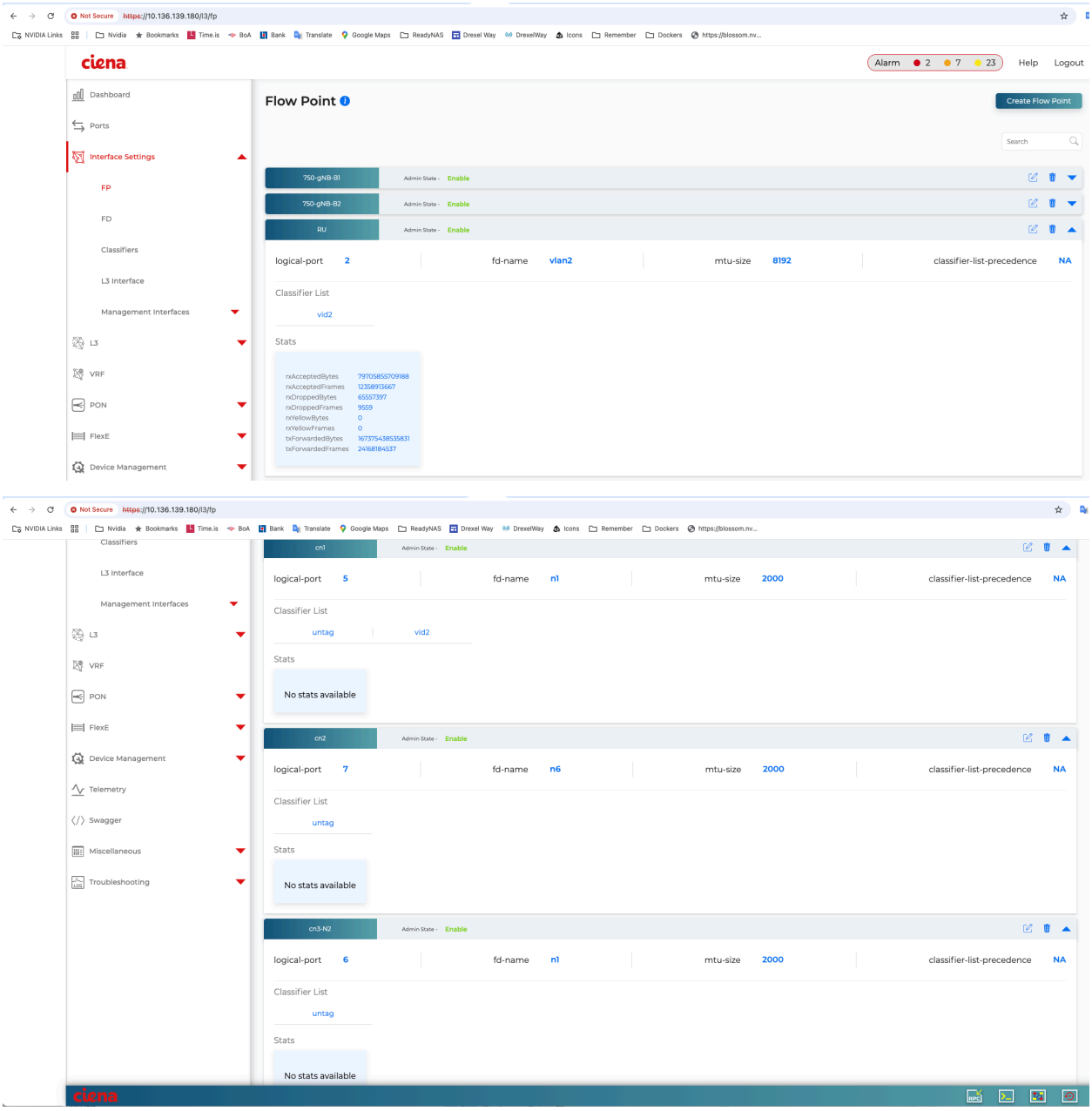
Note

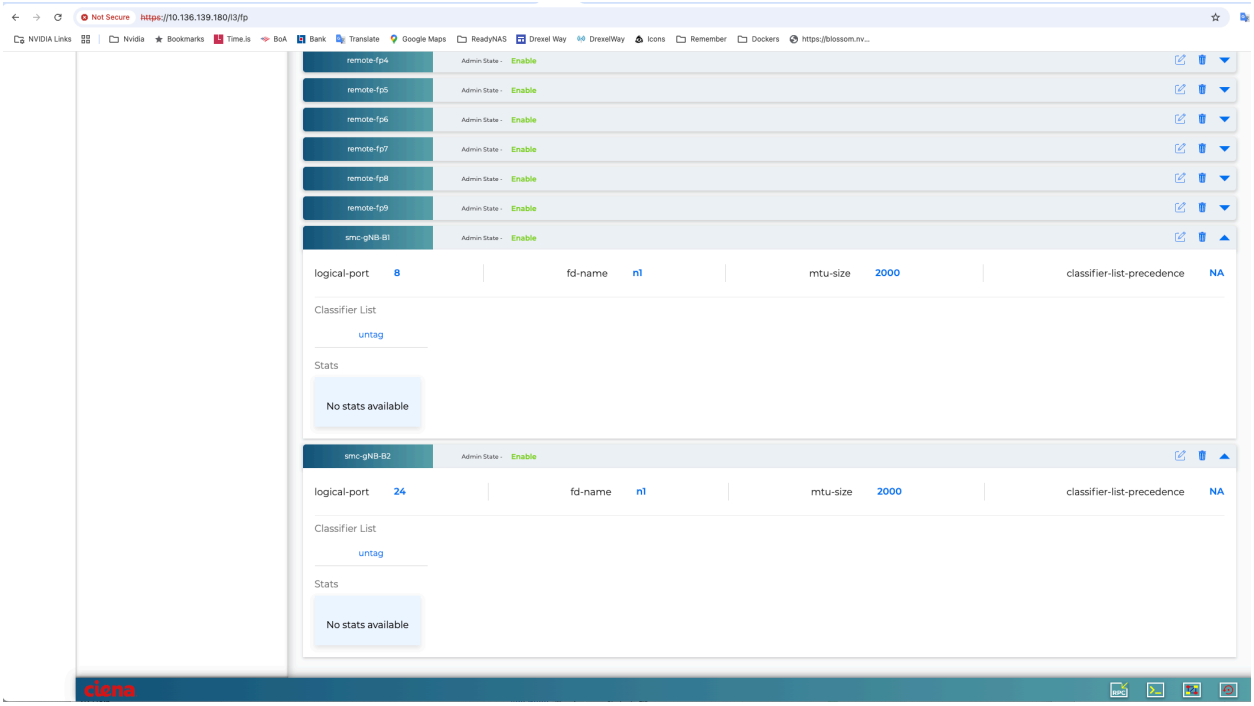
The switches in the screen shots have other equipment connected that is irrelevant.

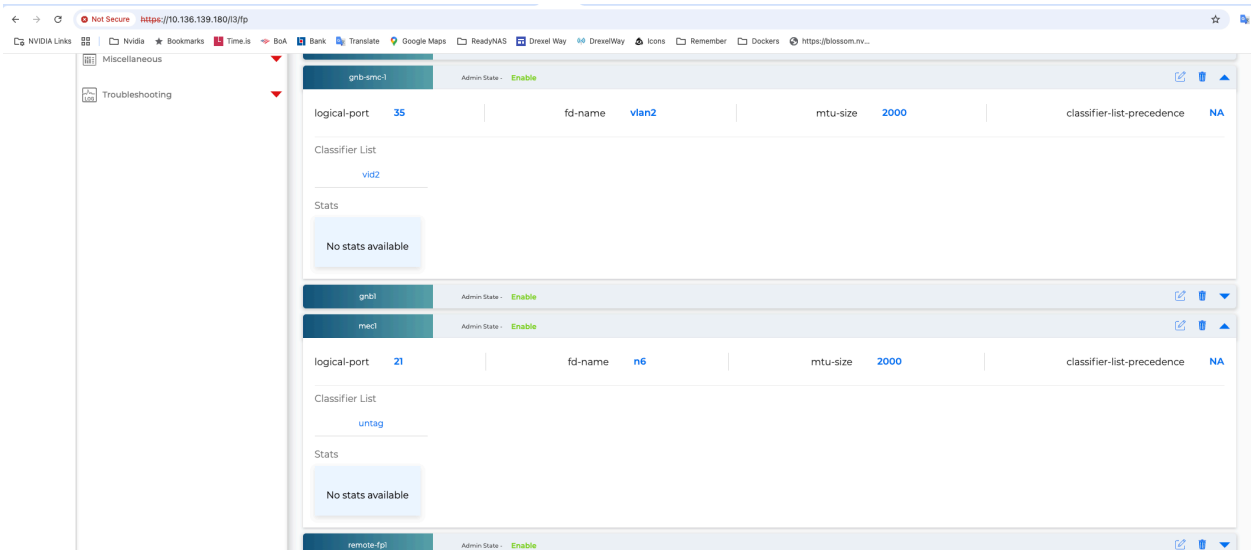
The switch has been setup so that it can be accessed both from a browser and using SSH.

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

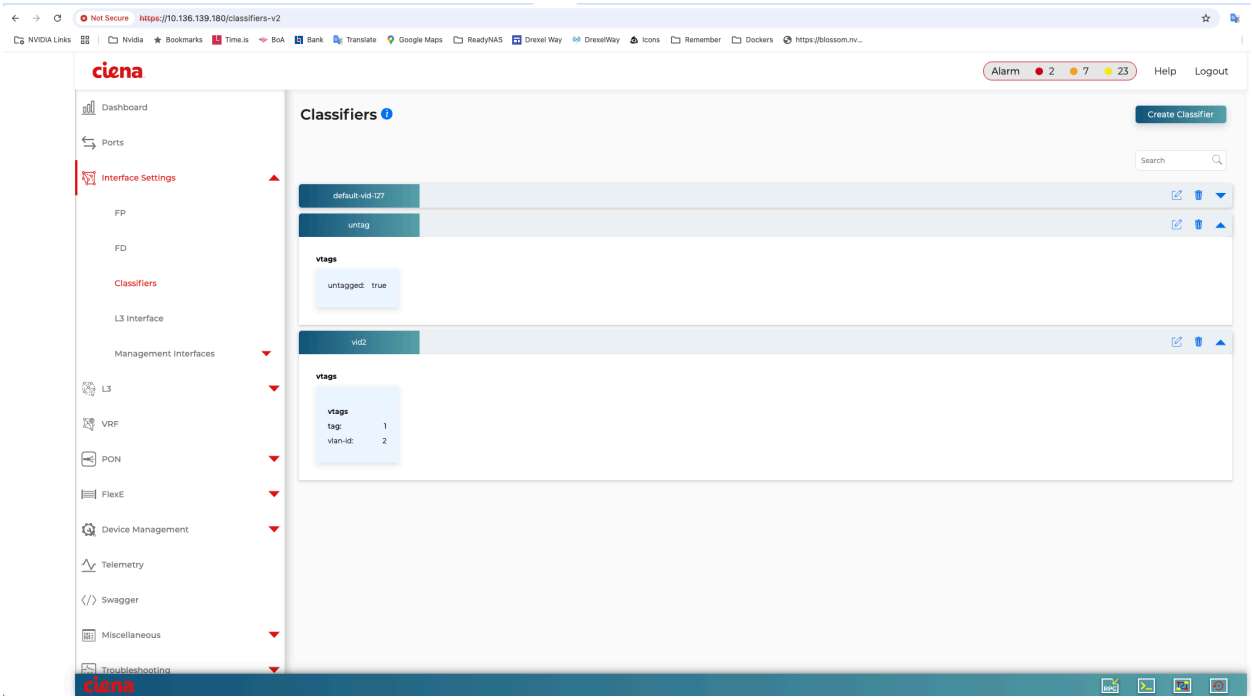
4.2.2.2.1 Create the Flow Points



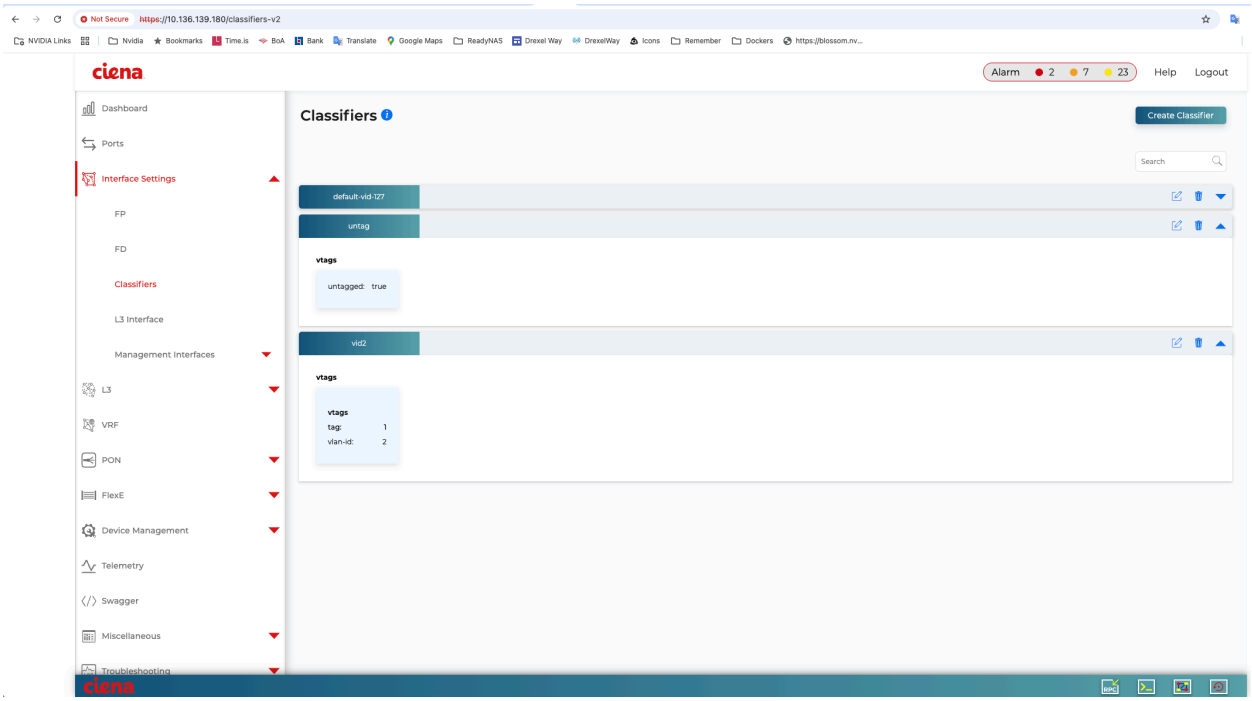

Not Secure https://10.136.139.180/l3/fp
remote-fp4
remote-fp5
remote-fp6
remote-fp7
remote-fp8
remote-fp9
smc-gNB-B1
logical-port 8
fd-name n1
mtu-size 2000
classifier-list-precedence NA
Classifier List
untag
Stats
No stats available
smc-gNB-B2
logical-port 24
fd-name n1
mtu-size 2000
classifier-list-precedence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


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

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

5164-2nd> show sync gnss almanac
```

(continues on next page)

(continued from previous page)

```
+------ 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 Foxconn 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.

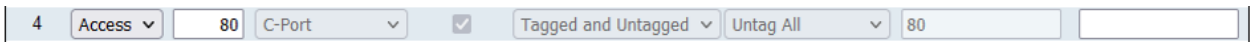


Fig. 2: Figure A - VLAN Setup

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

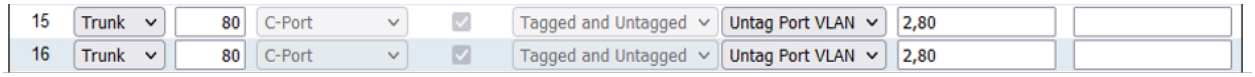
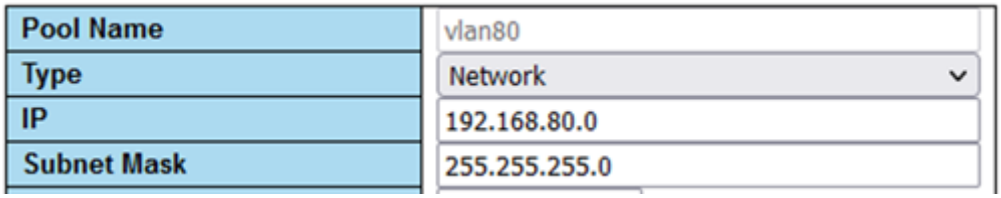


Fig. 3: Figure B - VLAN Setup

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

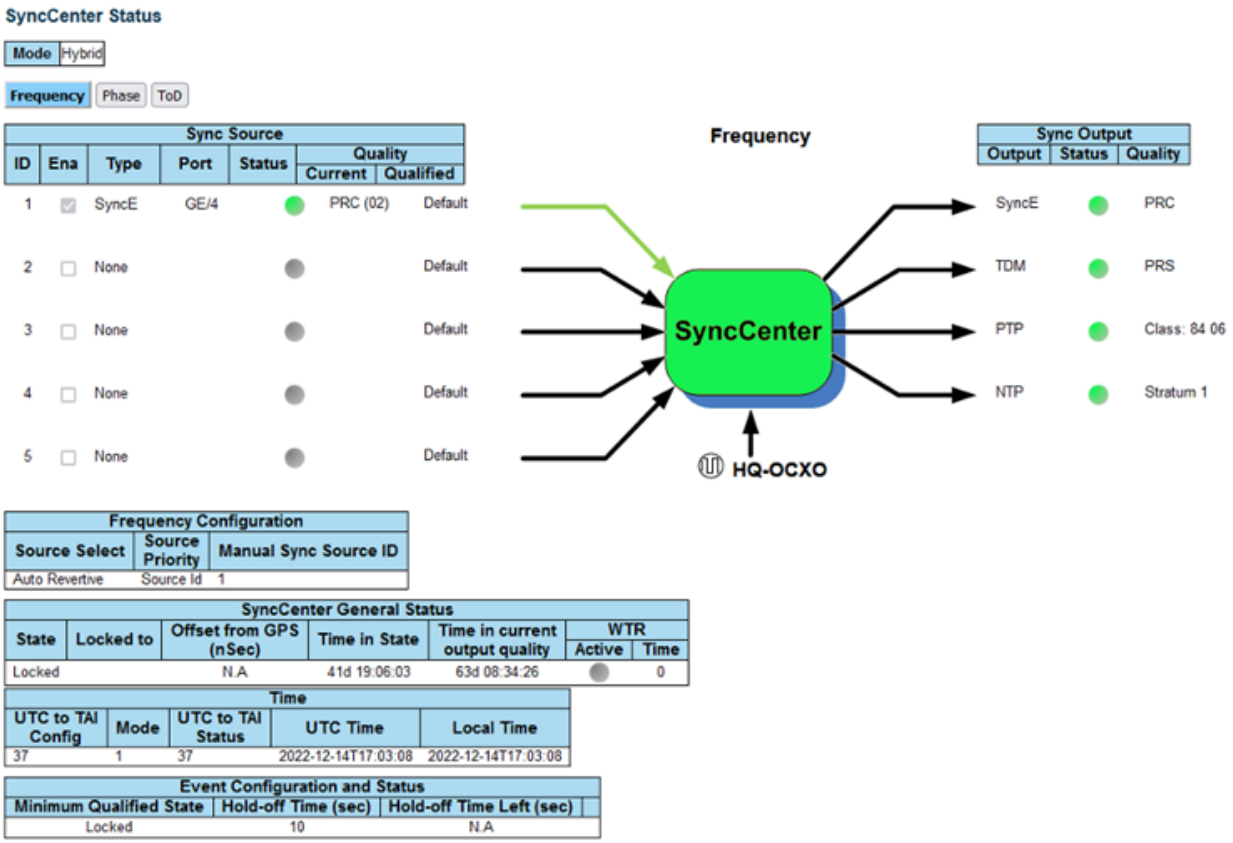
For the PTP setup, follow the Fibrolan PTP Boundary Clock Configuration guide and use the following settings:

- 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 PTP

These commands assume that PTP4L runs on the `ens6f0` NIC interface and uses CPU core 9. Core clash can cause problems, so if a different core is being used, it must not be used by L1 or L2+.

4.2.3.1 Verify Inbound PTP Packets

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

```
sudo tcpdump -i ens6f0 -c 5 | grep ethertype
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens6f1, 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:
```

4.2.3.2 Create ptp4l Configuration File

Paste these commands into the shell to create the three configuration files:

```
cat <<EOF | sudo tee /etc/ptp.conf
[global]
priority1 128
priority2 128
domainNumber 24
tx_timestamp_timeout 30
dscp_event 46
dscp_general 46
logging_level 6
verbose 1
use_syslog 0
logMinDelayReqInterval 1
[ens6f0]
logAnnounceInterval -3
announceReceiptTimeout 3
logSyncInterval -4
logMinDelayReqInterval -4
delay_mechanism E2E
network_transport L2
EOF

cat <<EOF | sudo tee /lib/systemd/system/ptp4l.service
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l

[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStart=/usr/bin/taskset -c 9 /usr/sbin/ptp4l -f /etc/ptp.conf

[Install]
WantedBy=multi-user.target
EOF
```

4.2.3.3 Create phc2sys Configuration File

```
# If more than one instance is already running, kill the existing
# PHC2SYS sessions.

# Command used can be found in /lib/systemd/system/phc2sys.service
# Update the ExecStart line to the following, assuming ens6f0 interface is used.
  sudo nano /lib/systemd/system/phc2sys.service

[Unit]
Description=Synchronize system clock or PTP hardware clock (PHC)
Documentation=man:phc2sys
After=ntpdate.service
Requires=ptp4l.service
After=ptp4l.service
```

(continues on next page)

(continued from previous page)

```
[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStart=/bin/sh -c "taskset -c 9 /usr/sbin/phc2sys -s /dev/ptp$(ethtool -T ens6f0 |
↪grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u 256"

[Install]
WantedBy=multi-user.target
```

4.2.3.4 Enable and Start phc2sys and ptp4l

After changing the configuration files, they need to be reloaded, enabled, and restarted. These services can be restarted if they don't sync.

```
sudo systemctl daemon-reload
sudo systemctl enable ptp4l.service
sudo systemctl enable phc2sys.service

sudo systemctl restart phc2sys.service ptp4l.service

# check that the service is active and has low rms value (<30):
systemctl status ptp4l.service phc2sys.service
  ptp4l.service - Precision Time Protocol (PTP) service
     Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset:
↪enabled)
     Active: active (running) since Tue 2023-05-09 13:21:12 UTC; 14s ago
       Docs: man:ptp4l
   Main PID: 6962 (ptp4l)
      Tasks: 1 (limit: 94588)
     Memory: 544.0K
     CGroup: /system.slice/ptp4l.service
             6962 /usr/sbin/ptp4l -f /etc/ptp.conf

May 09 13:21:17 aerial-rf-gb-gnb taskset[6962]: ptp4l[15552.609]: rms   15 max   32
↪freq   -639 +/-  25 delay   211 +/-   1
May 09 13:21:18 aerial-rf-gb-gnb taskset[6962]: ptp4l[15553.609]: rms   21 max   29
↪freq   -583 +/-  12 delay   210 +/-   1
May 09 13:21:19 aerial-rf-gb-gnb taskset[6962]: ptp4l[15554.609]: rms   11 max   21
↪freq   -576 +/-   8 delay   211 +/-   1
May 09 13:21:20 aerial-rf-gb-gnb taskset[6962]: ptp4l[15555.609]: rms    6 max   13
↪freq   -579 +/-   8 delay   211 +/-   1
May 09 13:21:21 aerial-rf-gb-gnb taskset[6962]: ptp4l[15556.609]: rms    4 max    7
↪freq   -578 +/-   6 delay   212 +/-   0
May 09 13:21:22 aerial-rf-gb-gnb taskset[6962]: ptp4l[15557.609]: rms    5 max   11
↪freq   -589 +/-   6 delay   213 +/-   1
May 09 13:21:23 aerial-rf-gb-gnb taskset[6962]: ptp4l[15558.609]: rms    6 max   12
↪freq   -593 +/-   8 delay   210 +/-   1
May 09 13:21:24 aerial-rf-gb-gnb taskset[6962]: ptp4l[15559.609]: rms    3 max    7
↪freq   -587 +/-   5 delay   211 +/-   1
May 09 13:21:25 aerial-rf-gb-gnb taskset[6962]: ptp4l[15560.609]: rms    5 max   12
↪freq   -582 +/-   7 delay   212 +/-   1
May 09 13:21:26 aerial-rf-gb-gnb taskset[6962]: ptp4l[15561.609]: rms    4 max    7
↪freq   -587 +/-   7 delay   213 +/-   1
```

(continues on next page)

(continued from previous page)

```
  phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
     Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset:
↪enabled)
     Active: active (running) since Tue 2023-05-09 13:21:12 UTC; 14s ago
       Docs: man:phc2sys
   Main PID: 6963 (phc2sys)
      Tasks: 1 (limit: 94588)
     Memory: 572.0K
     CGroup: /system.slice/phc2sys.service
             6963 /usr/sbin/phc2sys -a -r -n 24 -R 256 -u 256

May 09 13:21:17 aerial-rf-gb-gnb phc2sys[6963]: [15553.320] CLOCK_REALTIME rms   42
↪max   79 freq  +8240 +/- 368 delay  1762 +/-  16
May 09 13:21:18 aerial-rf-gb-gnb phc2sys[6963]: [15554.336] CLOCK_REALTIME rms   35
↪max   64 freq  +8091 +/- 303 delay  1754 +/-  13
May 09 13:21:19 aerial-rf-gb-gnb phc2sys[6963]: [15555.352] CLOCK_REALTIME rms   27
↪max   52 freq  +8218 +/- 224 delay  1752 +/-  13
May 09 13:21:20 aerial-rf-gb-gnb phc2sys[6963]: [15556.368] CLOCK_REALTIME rms   21
↪max   49 freq  +8153 +/- 152 delay  1758 +/-  16
May 09 13:21:21 aerial-rf-gb-gnb phc2sys[6963]: [15557.384] CLOCK_REALTIME rms   17
↪max   39 freq  +8149 +/- 125 delay  1761 +/-  16
May 09 13:21:22 aerial-rf-gb-gnb phc2sys[6963]: [15558.400] CLOCK_REALTIME rms   14
↪max   33 freq  +8185 +/- 101 delay  1750 +/-  14
May 09 13:21:23 aerial-rf-gb-gnb phc2sys[6963]: [15559.416] CLOCK_REALTIME rms   12
↪max   32 freq  +8138 +/-  63 delay  1752 +/-  13
May 09 13:21:24 aerial-rf-gb-gnb phc2sys[6963]: [15560.431] CLOCK_REALTIME rms   11
↪max   43 freq  +8171 +/-  54 delay  1756 +/-  15
May 09 13:21:25 aerial-rf-gb-gnb phc2sys[6963]: [15561.447] CLOCK_REALTIME rms   10
↪max   32 freq  +8163 +/-  38 delay  1762 +/-  16
May 09 13:21:26 aerial-rf-gb-gnb phc2sys[6963]: [15562.463] CLOCK_REALTIME rms    9
↪max   23 freq  +8162 +/-  17 delay  1761 +/-  16
```

4.2.3.5 Disable NTP

Use these commands to turn off NTP:

```
sudo timedatectl set-ntp false
timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
           Universal time: Thu 2022-02-03 22:30:58 UTC
                 RTC time: Thu 2022-02-03 22:30:58
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
```

4.2.3.6 Verify System Clock Synchronization

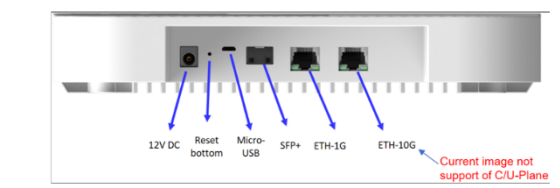
Make NTP inactive and synchronize the system clock:

```
timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
           Universal time: Thu 2022-02-03 22:30:58 UTC
                 RTC time: Thu 2022-02-03 22:30:58
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: inactive
          RTC in local TZ: no
```

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

Tip

There is a *tutorial video* for setting up the Foxconn O-RU.

Foxconn RPQN-7801E	Connections and Settings
 **Antenna port number** 	Connections: ▶ **10SFP**: C/U plane (will support S/M plane after firmware upgrade) ▶ **1G RJ45**: S/M plane ▶ **10G RJ45**: POE only ▶ **Micro-USB**: USB to serial for debugging (115200, 8, 1, none, flow control off) VIAVI GrandMaster settings: ▶ **PTP timing port**: Disable VLAN ▶ **Two steps**: OFF ▶ **Domain number**: 24 (needs to configured on O-RU) ▶ IPv4, Unicast, etc. `/home/root/sdcard/RRHconfig_xran.xml`: ▶ `RRH_PTPV2_GRAND_MASTER_IP = 20.0.0.8` ▶ `RRH_PTPV2_SUB_DOMAIN_NUM = 24` ▶ C/U plane VLAN tag ▶ `RRH_LO_FREQUENCY_KHZ = 3750000`

4.2.4.1 Configure VLAN and IP Address on the gNB Server

1. Add these commands to the server startup script (`/etc/rc.local`) so they are automatically run on reboot.
2. Configure these settings on the fronthaul port.
3. You must use IP addresses that do not match those in the example below:

```
sudo ip link add link ens6f0 name ens6f0.2 type vlan id 2
sudo ip addr add 169.254.1.103/24 dev ens6f0.2
```

(continues on next page)

(continued from previous page)

```
sudo ip link set up ens6f0.2
```

4.2.4.2 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
```

4.2.4.3 Update O-RU Configuration

Note

If you are using the CBRS O-RU (Foxconn RPQN-4800E), 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
```

(continues on next page)

(continued from previous page)

```
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

In Foxconn firmware version 2.6.9, the configuration file is located in `/home/root/sdcard`.

Note

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

Note

If you're using the CBRS O-RU (Foxconn RPQN-4800E), the above parameters should be modified as follows:

- n78:

  `RRH_LO_FREQUENCY_KHZ = 3750000, 0`

- n48 (CBRS):

  `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
```

4.3. Part 3. Configure the gNB Server

To install the Aerial tools, follow the cuBB installation guide; refer to the *Software Manifest* for a link to the cuBB documentation.

4.3.1. Configure the gNB Server - Gigabyte Edge E251-U70

To install the Aerial cuBB tools, follow the Installing Tools on Gigabyte Edge E251-U70 instructions.

The ARC-OTA thread-to-core assignment functionality is different from a standard Aerial installation. Layer 1 threads need to be isolated in a monolithic block and have been moved, with the rest left to layers 2 and higher: Cores 2-6 are used as cuPHY worker cores; core 7 is used for the cuPHY `lowprio` thread; core 8 is used for the cuPHY timer thread; and core 9 is used for PTP and PHC2SYS.

4.3.1.1 Configure Linux Kernel Command-Line for ARC-OTA

To set kernel command-line parameters, edit the `GRUB_CMDLINE_LINUX_DEFAULT` parameter in the grub file `/etc/default/grub` and modify the following parameters.

Note

The following kernel parameters are optimized for the Gigabyte server with 24 cores Xeon Gold 6240R and 96GB memory. For ARC-OTA, typically it is optimal to configure the gNB to isolate cores 2 to 10 for Aerial, and leave the other cores for OAI L2+.

```
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=2-10
rcu_nocb_poll
nosoftlockup
iommu=off
intel_iommu=off
irqaffinity=0-1,22-23
```

To automatically append the grub file with these changes, use this command:

```
sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT="[^"]*/& 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=2-10 rcu_nocb_poll nosoftlockup iommu=off intel_iommu=off irqaffinity=0-1,
↪22-23/' /etc/default/grub
```

4.3.1.2 Apply the Changes

1. Use the following commands to apply the changes and reboot to load the kernel.

   ```
   sudo update-grub
   sudo reboot
   ```

2. After rebooting, enter the following command to check whether the system has booted into the low-latency kernel:

   ```
   uname -r
    5.15.0-1042-nvidia-lowlatency
   ```

3. Enter this command to check that the kernel command-line parameters are configured correctly:

   ```
   cat /proc/cmdline
   BOOT_IMAGE=/vmlinuz-5.15.0-1042-nvidia-lowlatency root=/dev/mapper/ubuntu-
   ↪-vg-ubuntu--lv 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=2-10 rcu_nocb_poll nosoftlockup iommu=off intel_iommu=off
   ↪irqaffinity=0-1,22-23
   ```

4.3.1.3 Change Core for ptp4l and phc2sys

Edit the `/lib/systemd/system/ptp4l.service` file as follows:

```
ExecStart=taskset -c 9 /usr/sbin/ptp4l -f /etc/ptp.conf
```

Edit the `/lib/systemd/system/phc2sys.service` file as follows:

```
ExecStart=/bin/sh -c "taskset -c 9 /usr/sbin/phc2sys -s /dev/ptp$(ethtool -T ens6f0 |
↪grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u 256"
```

4.3.2. Configure gNB Server - Dell R750

To install the Aerial cuBB tools, follow the Dell R750 cuBB installation guide.

The ARC-OTA thread-to-core assignment functionality is different from a standard Aerial installation. Layer 1 threads need to be isolated in a monolithic block and have been moved, with the rest left to layers 2 and higher. Cores [5,7,9,11,13] are used as cuPHY workers cores. Core 17 is used for the cuPHY `lowprio` thread; core 15 is used for the cuPHY timer thread; and core 19 is used for PTP and PHC2SYS.

4.3.2.1 Configure the Linux Kernel Command Line for ARC-OTA

To set kernel command-line parameters, modify the following parameters under the `GRUB_CMDLINE_LINUX_DEFAULT` parameter in the grub file (`/etc/default/grub`).

```
pci=realloc=off
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
rcu_nocb_poll
nosoftlockup
iommu=off
irqaffinity=0-3,44-47
isolcpus=5,7,9,11,13,15,17,19,21
noht
```

To automatically append the grub file with these changes, use this command:

```
sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT="[^"]*/& pci=realloc=off 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 rcu_nocb_poll nosoftlockup iommu=off irqaffinity=0-3,44-47 isolcpus=5,7,9,
↪11,13,15,17,19,21 noht/' /etc/default/grub
```

4.3.2.2 Apply the Changes and Load the Kernel

1. Use the following commands to apply the command-line changes and reboot the system.

   ```
   sudo update-grub
   sudo reboot
   ```

2. After rebooting, use the following command to check whether the system has booted into the low-latency kernel:

   ```
   uname -r
    5.15.0-1042-nvidia-lowlatency
   ```

3. Enter this command to check that the kernel command-line parameters are configured correctly:

   ```
   cat /proc/cmdline
   BOOT_IMAGE=/vmlinuz-5.15.0-1042-nvidia-lowlatency root=/dev/mapper/ubuntu-
   ↪-vg-ubuntu--lv ro pci=realloc=off 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 rcu_nocb_poll nosoftlockup iommu=off irqaffinity=0-3,44-47
   ↪isolcpus=5,7,9,11,13,15,17,19,21 noht
   ```

4.3.2.3 Change the Core for ptp4l and phc2sys

Edit the `/lib/systemd/system/ptp4l.service` file:

```
ExecStart=taskset -c 19 /usr/sbin/ptp4l -f /etc/ptp.conf
```

Edit the `/lib/systemd/system/phc2sys.service` file:

```
ExecStart=/bin/sh -c "taskset -c 19 /usr/sbin/phc2sys -s /dev/ptp$(ethtool -T
↪enp204s0f1np1 | grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u
↪256"
```

4.3.3. Configure gNB Server - SMC Grace Hopper MGX

To install the Aerial cuBB tools, follow the same steps as the Grace Hopper MGX installation guide.

The ARC-OTA thread-to-core assignment functionality is different from a standard Aerial installation. Layer 1 threads need to be isolated in a monolithic block and have been moved, with the rest left to layers 2 and higher. Cores [5,7,9,11,13] are used as cuPHY worker cores. Core 17 is used for the cuPHY `lowprio` thread; core 15 is used for the cuPHY timer thread; and core 41 is used for PTP and PHC2SYS.

4.3.3.1 Configure the Linux Kernel Command Line for ARC-OTA

To set kernel command-line parameters, edit the `GRUB_CMDLINE_LINUX` parameter in the grub file `/etc/default/grub.d/cmdline.cfg` and append or update the parameters described below. The following kernel parameters are optimized for GH200. To automatically append the grub file with these parameters, use this command:

```
cat <<"EOF" | sudo tee /etc/default/grub.d/cmdline.cfg
GRUB_CMDLINE_LINUX="$GRUB_CMDLINE_LINUX pci=realloc=off pci=pcie_bus_safe default_
↪hugepagesz=512M hugepagesz=512M hugepages=32 tsc=reliable processor.max_cstate=0
↪audit=0 idle=poll rcu_nocb_poll nosoftlockup irqaffinity=0 isolcpus=managed_irq,
↪domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 earlycon module_blacklist=nouveau acpi_
↪power_meter.force_cap_on=y numa_balancing=disable init_on_alloc=0 preempt=none"
EOF
```

4.3.3.2 Apply the Changes and Load the Kernel

1. Use the following commands to apply the command-line changes and reboot the system.

   ```
   sudo update-grub
   sudo reboot
   ```

2. After rebooting, use the following command to check whether the system has booted into the low-latency kernel:

   ```
   uname -r
   6.2.0-1012-nvidia-64k
   ```

3. Enter this command to check that the kernel command-line parameters are configured correctly:

```
cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-6.2.0-1012-nvidia-64k root=/dev/mapper/ubuntu--vg-
↪ubuntu--lv ro pci=realloc=off pci=pcie_bus_safe default_hugepagesz=512M
↪hugepagesz=512M hugepages=32 tsc=reliable processor.max_cstate=0
↪audit=0 idle=poll rcu_nocb_poll nosoftlockup irqaffinity=0
↪isolcpus=managed_irq,domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 earlycon
↪module_blacklist=nouveau acpi_power_meter.force_cap_on=y numa_
↪balancing=disable init_on_alloc=0 preempt=none
```

4.3.3.3 Change the Core for ptp4l and phc2sys

Edit the `/lib/systemd/system/ptp4l.service` file:

```
ExecStart=taskset -c 41 /usr/sbin/ptp4l -f /etc/ptp.conf
```

Edit the `/lib/systemd/system/phc2sys.service` file:

```
ExecStart=/bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -T
↪aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u 256"
```

4.4. Part 4. Install ARC-OTA Using NVIDIA SDK Manager

You can use NVIDIA SDK Manager (SDKM) to install directly on the target machine or from a remote machine. SDKM uses SSH to connect to the server, where it installs the gNB.

Important

The SDKM client cannot run on ARM, so SDKM installation is not supported on SMC-GH systems.

Steps to install ARC-OTA using SDK Manager can be found in the SDK Manager documentation

You can download the installer at https://developer.download.nvidia.com/sdkmanager/redirects/sdkmanager-deb.html

4.4.1. Prerequisites for Installing gNB with SDK Manager

Use the Aerial cuBB Installation Guide for the following steps:

1. Configure BIOS Settings
2. Install Ubuntu 22.04 Server
3. Install the Low-Latency Kernel
4. Configure Linux Kernel Command-line

4.4.2. Post-Installation Steps

1. Configure OAI L2.

   IP addresses of the CN and the gNB must be configured using one of the following:

   - Update gnb_cn5g_network_config.ini file with the actual network setting values. Then run gnb_apply_network_config.sh to apply the settings
   - Or follow the ARC-OTA installation guide steps to do it manually

2. Jump to **Part 5 – Validate the Setup**.

   Follow the steps in the ARC-OTA Installation Guide.

4.5. Part 5. Validate the Setup

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

4.5.1. 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.

4.5.2. 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
```

(continues on next page)

(continued from previous page)

```
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
```

4.5.2.1 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.

4.5.3. 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.

4.5.3.1 Start CN5G Core Network

Use the following commands to start the CN5G Core Network.

```
sudo sysctl net.ipv4.conf.all.forwarding=1
sudo iptables -P FORWARD ACCEPT

cd ~/oai-cn5g
docker compose up -d
```

4.5.3.1.1 Start the CN5G Edge Application

After the CN5G is started, use the `oai-ext-dn` container to run IPERF.

```
docker exec -it oai-ext-dn /bin/bash
```

4.5.3.2 Start Aerial cuBB on the gNB

Execute the following command to set up the cuBB container.

```
# Run on host: start a docker terminal
docker exec -it $AERIAL_CUBB_CONTAINER /bin/bash
```

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.

```
cd /opt/nvidia/cuBB
export cuBB_SDK=$(pwd)
mkdir build && cd build
cmake ..
make -j
```

The Aerial cuPHY configuration files are located in the `/opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config` directory.

For ARC-OTA 1.5.1, the setup has been validated with the following configuration files:

- `cuphycontroller_P5G_FXN.yaml` for the Gigabyte
- `servercuphycontroller_P5G_FXN_R750.yaml` for the Dell R750 server
- `cuphycontroller_P5G_FXN_R750.yaml` for the Supermicro GH200 server.

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. The following example applies to the Gigabyte server.

```
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
```

When the build is done and the configuration files are updated, exit the container. Run the following to commit the changes to a new image. The name of the image will be used later in the Docker Compose configuration.

```
docker commit $AERIAL_CUBB_CONTAINER cubb-build:24-1
```

4.5.3.3 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 container and add it to the OAI build files. Execute the following to create the `nvipc_src.<data>.tar.gz` file.

```
docker exec -it $AERIAL_CUBB_CONTAINER ./cuPHY-CP/gt_common_libs/pack_nvipc.sh
docker cp $AERIAL_CUBB_CONTAINER:/opt/nvidia/cuBB/cuPHY-CP/gt_common_libs/nvipc_src.
↪<date>.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.

4.5.3.4 Build gNB Docker Image

In ARC-OTA 1.5.1, 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.

4.5.3.5 Pre-build Steps for OAI L2

1. When building OAI L2 for Aerial CUDA-Accelerated RAN 24-1, remove line 50 and 51 in the `docker/Dockerfile.gNB.aerial.ubuntu20` file before building. (ARC-OTA 1.5. includes the `OAI 2024.w21` tag and Aerial CUDA-Accelerated RAN 24-1). This will be merged to the OAI `develop` branch in the future.
2. In the same file, add `moreutils` on line 79.

```
moreutils \
```

3. Remove the pinning of `p7_thread` in the OAI FAPI integration (`nfapi/oai_integration/aerial/fapi_vnf_p5.c`). To do so, remove lines 59-64 (this change will be merged to the OAI `develop` branch in the future). Core 8 is occupied by L1 on the Gigabyte server. On Dell R750, this core is free, but gNB is usually started on numa 0 with odd cores.

```
//CPU_SET(8, &cpuset);
//s = pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t), &cpuset);
//if (s != 0)
//  printf("failed to set afinity\n");
```

4. Ensure the the FAPI polling thread is run on an isolated core by modifying line 609 of the `nfapi/oai_integration/aerial/fapi_nvIPC.c` file.

   On the Gigabyte server, Dell R750 server, and SMC GH200 server, use core 21:

```
stick_this_thread_to_core(21)
```

5. There is also a bug regarding PDU length truncation in the `OAI 2024.w21` tag used with ARC 1.5. This is already fixed on the `develop` branch of OAI. The current ARC release has been verified by applying this MR on top of the 2024.w21 `OAI 2024.w21` tag.

```
cd ~/openairinterface5g/
wget https://gitlab.eurecom.fr/oai/openairinterface5g/-/merge_requests/2797.patch
git apply -3 2797.patch
```

Use the below commands to build the OAI L2 code:

```
cd ~/openairinterface5g/
docker build . -f docker/Dockerfile.base.ubuntu20 --tag ran-base:latest
docker build . -f docker/Dockerfile.gNB.aerial.ubuntu20 --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 then 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.

4.5.3.6 Running gNB with Docker Compose

ARC-OTA 1.5.1 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 is located in the following file path:

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

Before starting the gNB, you will need to update the `docker-compose.yaml` file.

1. On line 27, switch the cuBB image to the image that was built, committed, and tagged in the previous steps.

   ```
   image: cubb-build:24-1
   ```

2. On line 30, add `sudo` to the command.

   ```
   command: bash -c "sudo rm -rf /tmp/phy.log && sudo chmod +x /opt/nvidia/cuBB/
   ↪aerial_l1_entrypoint.sh && /opt/nvidia/cuBB/aerial_l1_entrypoint.sh"
   ```

3. On line 37, if you follow the guide, change the `cpu_set` for the OAI container as follows.

   a. Gigabyte server:

   ```
   cpu_set: "13-20"
   ```

   b. Dell R750 server:

   ```
   cpu_set: "23,25,27,29,31,33,35,37"
   ```

   c. SMC-GH server:

   ```
   cpuset: "11-18"
   ```

4. On line 60, add a volume for the `oai.log` file.

   ```
   - /var/log/aerial:/var/log/aerial
   ```

5. On line 61 (after the volumes), add a command for executing the `nr-softmodem`. This will replace the default command that is integrated in the docker image. This new command will set the priority and create an `oai.log` file with time stamp.

```
command: bash -c "chrt -f 99 /opt/oai-gnb/bin/nr-softmodem -O /opt/oai-gnb/etc/
↪gnb.conf | ts | tee /var/log/aerial/oai.log"
```

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

```
$ docker image ls
REPOSITORY                                                    TAG
↪                                    IMAGE ID       CREATED          SIZE
cubb-build                                                    24-1
↪                                    be7a5a94f2d3   10 seconds ago   26GB
nvcr.io/qhrjhjrvlsbu/aerial-cuda-accelerated-ran              24-1-cubb
↪                                    a66cf2a45fad   2 months ago    25.5GB
oai-gnb-aerial                                                latest
↪                                    0856b9969f42   3 weeks ago      4.88GB
ran-base                                                      latest
↪                                    c5d060d23529   3 weeks ago      2.42GB
```

Docker Compose will start containers running cuBB and OAI L2+. The Docker Compose script includes an entry-point script for cuBB, which you need to modify before running. The script points at the `cuphycontroller_xxx.yaml` configuration that you want to run. This script is located in the following file path:

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

Before running Docker Compose, update the `aerial_l1_entrypoint.sh` file.

1. The latest ARC-OTA release does not have to build `gdrcopy`.

```
# cd "$cuBB_Path"/cuPHY-CP/external/gdrcopy/ || exit 1
# ./insmod.sh
```

2. In the latest ARC-OTA release, you are not root by default. Add `sudo` to the following, including `P5G_FXN_R750` if you are using a Dell R750 or P5G_FXN if you are using .

```
sudo -E "$cuBB_Path"/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf
↪P5G_FXN_R750
```

3. Before running the Docker Compose script, you also need to add root for some commands that are run before starting Aerial cuPHY Layer 1. To do so, edit the `ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml` file.

```
command: bash -c " sudo rm -rf /tmp/phy.log && sudo chmod +x /opt/nvidia/cuBB/
↪aerial_l1_entrypoint.sh && /opt/nvidia/cuBB/aerial_l1_entrypoint.sh"
```

   Also, OAI L2 must point to the correct configuration by editing the following row in the `ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml` file. All L2 configurations can be found in `targets/PROJECTS/GENERIC-NR-5GC/CONF`.

```
- ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb-vnf.sa.band78.273prb.aerial.
↪conf:/opt/oai-gnb/etc/gnb.conf
```

4. On line 37 in `ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml`, specify the cores to use for OAI l2.

   a. Gigabyte server:

```
cpuset: "13-20"
```

b. Dell R750 server:

```
cpuset: "23,25,27,29,31,33,35,37"
```

c. SMC-GH server:

```
cpuset: "11-18"
```

You can now run ARC-OTA 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
// tail -f /var/log/aerial/oai.log
```

4.5.3.7 Connecting the Commercial UE to the 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.

4.5.3.7.1 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.

4.5.3.8 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.

4.5.3.8.1 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
```

4.5.3.8.2 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
```

4.5.3.8.3 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 stop $OAI_GNB_CONTAINER
docker rm $OAI_GNB_CONTAINER
```

Note

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

4.6. ARC-OTA Configuration App Note (Step-by-Step Debug Commands)

4.6.1. Setup Aerial CUDA-Accelerated RAN

In the installation guide for cuBB, find the Aerial CUDA-Accelerated RAN section and follow the instructions.

4.6.2. Running the cuBB Docker Container

```
export GPU_FLAG="--gpus all"
export cuBB_SDK=/opt/nvidia/cuBB
#Name of your docker container
export AERIAL_CUBB_CONTAINER=cuBB_$USER
#Docker image downloaded from NGC
export AERIAL_CUBB_IMAGE=nvcr.io/qhrjhjrvlsbu/aerial-cuda-accelerated-ran:24-1-cubb

sudo usermod -aG docker $USER

docker run --detach --privileged \
   -it $GPU_FLAG --name $AERIAL_CUBB_CONTAINER \
   --hostname c_aerial_$USER \
   --add-host c_aerial_$USER:127.0.0.1 \
   --network host \
   --shm-size=4096m \
   -e cuBB_SDK=$cuBB_SDK \
   -w $cuBB_SDK \
   -v $(echo ~):$(echo ~) \
   -v /dev/hugepages:/dev/hugepages \
   -v /usr/src:/usr/src \
   -v /lib/modules:/lib/modules \
   -v ~/share:/opt/cuBB/share \
   --userns=host \
   --ipc=host \
   -v /var/log/aerial:/var/log/aerial \
   $AERIAL_CUBB_IMAGE

docker exec -it $AERIAL_CUBB_CONTAINER bash
```

For installation instructions, see the Aerial cuBB Installation Guide, in the link above.

Since the cuBB 23-4 release, the necessary testvectors for running OTA are already included. For running the Aerial E2E test with ru-emulator and test mac, follow the Aerial CUDA-Accelerated RAN documentation for generating the required testvectors.

4.6.3. Setup OAI gNB

4.6.3.1 Clone the gNB Source Code

Clone the OpenAirInterface5G repository.

```
git clone --branch  2024.w21+ARC1.5 https://gitlab.eurecom.fr/oai/openairinterface5g.
↪git ~/openairinterface5g

cd openairinterface5g
```

4.6.3.2 gNB Configuration File

Update the configuration of OAI L2. The configuration is located here.

The L1 configuration to use is included in the latest Aerial release image, and will differ depending on the gNB server you are using:

- **Gigiabyte**: Use the following file:

```
cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_FXN.yaml
```

- **SMC-GH**: Use the same configuration file as the Gigabyte server, ensuring that the cores are isolated as described in *Configure gNB Server - SMC Grace Hopper MGX*:

```
cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_FXN.yaml
```

- **Dell R750**: Use the following file:

```
cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_FXN_R750.yaml
```

4.6.4. Setup OAI CN5G

Do the iptables setup below after every system reboot, or make this setup permanent in your Ubuntu system configuration.

```
On CN5G server, configure it to allow the traffic coming in by
adding this rule to iptables:

# On CN5G, upon startup:

sudo sysctl net.ipv4.conf.all.forwarding=1
sudo iptables -P FORWARD ACCEPT
```

Install the Core Network by following the Gitlab steps for setting up OAI CN5G.

To run the correct configuration for ARC-OTA, replace section 2.2 and 2.3 OAI CN5G configuration files with the following:

```
# Get openairinterface5g source code
git clone --branch 2024.w21+ARC1.5 https://gitlab.eurecom.fr/oai/openairinterface5g.
↪git ~/openairinterface5g
cd ~/openairinterface5g
cp -rT ~/openairinterface5g/doc/tutorial_resources/oai-cn5g ~/oai-cn5g
```

The user configurable configuration files are:

- ~/oai-cn5g/database/oai_db.sql

4.6.5. Configuring OAI gNB and CN5G

For the purpose of understanding which address is what in the example configuration setting and commands below, we will assume the gNB and CN5G servers have these interface names and IP addresses.

CN5G Server

```
eno1: 10.31.66.x            = SSH management port for terminal
eno2: 169.254.200.6         = BH connection on SFP switch for gNB-CN5G traffic
```

gNB Server

```
eno1:     10.31.66.x        = SSH management port for terminal
ens6f0:   b8:ce:f6:4e:75:40 = FH MAC address
ens6f0.2: 169.254.1.105     = FH IP address
ens6f1:   169.254.200.5     = BH connection SFP switch for gNB-CN5G traffic
```

gNB to set static route

On the gNB server, add this static route for a path to the CN5G server. Apply this route after each server power-on.

```
Syntax:
sudo ip route add 192.168.70.128/26 via <CN5G IP> dev <gNB interface for CN5G>

Example:
sudo ip route add 192.168.70.128/26 via 169.254.200.6 dev ens6f1
```

gNB to set the CN5G server to uses for AMF

Edit the used gNB configuration file. The configuration file for Aerial can be found here:

```
~/openairinterface5g/targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb-vnf.sa.band78.273prb.
↪aerial.conf
```

Update the Docker compose file to mount the updated file.

Below is an example with lab-specific network parameters. Your IP address and interface names may differ.

```
GNB_INTERFACE_NAME_FOR_NG_AMF = "ens6f1";       # gNB side interface name of the SFP
↪port toward CN (was eno1)
GNB_IPV4_ADDRESS_FOR_NG_AMF = "169.254.200.5";  # gNB side IP address of interface
↪above (was 172.21.16.130)
GNB_INTERFACE_NAME_FOR_NGU = "ens6f1";          # gNB side interface name of the SFP
↪port toward CN (was eno1)
GNB_IPV4_ADDRESS_FOR_NGU = "169.254.200.5";     # Same IP as GNB_IPV4_ADDRESS_FOR_NG_
↪AMF above (was 172.21.16.130)
```

4.6.6. Running CN5G

4.6.6.1 To start CN5G

```
docker-compose up -d
```

For SMC-GH, you will need to patch some of the CN components. Contact aerial-info@nvidia.com for the patches and build instructions. Configuration files can be found here.

4.6.6.2 To Stop CN5G

```
docker-compose down
```

4.6.6.3 To monitor CN5G logs while running

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

4.6.6.4 To capture PCAPs

```
docker exec -it oai-amf /bin/bash
apt update && apt install tcpdump -y
tcpdump -i any -w /tmp/amf.pcap
```

Then copy the pcap out from the container.

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

4.6.7. Example Screenshot of Starting CN5G

```
aerial@aerial-rf-r630:~/oai-cn5g$ docker compose up -d
[+] Building 0.0s (0/0)
[+] Running 11/11
✓ Network demo-oai-public-net  Created
 ↪                    0.1s
✓ Container oai-nrf            Started
 ↪                    0.7s
✓ Container mysql              Started
 ↪                    0.7s
✓ Container asterisk-ims       Started
 ↪                    0.7s
✓ Container oai-udr            Started
 ↪                    0.9s
✓ Container oai-udm            Started
 ↪                    1.2s
✓ Container oai-ausf           Started
 ↪                    1.5s
✓ Container oai-amf            Started
```

(continues on next page)

(continued from previous page)

```
 ↪                    1.7s
✓ Container oai-smf           Started
 ↪                    2.0s
✓ Container oai-spgwu-tiny    Started
 ↪                    2.3s
✓ Container oai-ext-dn        Started
 ↪                    2.6s
```

```
aerial@aerial-rf-r630:~/oai-cn5g$ docker ps
CONTAINER ID   IMAGE                                          COMMAND
 ↪CREATED              STATUS                          PORTS
 ↪NAMES
d5af4f51c393   oaisoftwarealliance/trf-gen-cn5g:latest     "/bin/bash -c ' ip r…"
 ↪About a minute ago   Up About a minute (healthy)
 ↪oai-ext-dn
a9b2d18c7f77   oaisoftwarealliance/oai-spgwu-tiny:v1.5.1   "python3 /openair-sp…"
 ↪About a minute ago   Up About a minute (healthy)   2152/udp, 8805/udp
 ↪oai-spgwu-tiny
b61c383f9e60   oaisoftwarealliance/oai-smf:v1.5.1          "python3 /openair-sm…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp, 8080/tcp, 8805/udp
 ↪oai-smf
3681b1048c53   oaisoftwarealliance/oai-amf:v1.5.1          "python3 /openair-am…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp, 9090/tcp, 38412/sctp
 ↪oai-amf
c602f7cb1c67   oaisoftwarealliance/oai-ausf:v1.5.1         "python3 /openair-au…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp
 ↪oai-ausf
752acae83ac0   oaisoftwarealliance/oai-udm:v1.5.1          "python3 /openair-ud…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp
 ↪oai-udm
4bf281d08229   oaisoftwarealliance/oai-udr:v1.5.1          "python3 /openair-ud…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp
 ↪oai-udr
33aa959be775   mysql:8.0                                   "docker-entrypoint.s…"
 ↪About a minute ago   Up About a minute (healthy)   3306/tcp, 33060/tcp
 ↪mysql
5d22e4745d00   asterisk-ims:latest                         "asterisk -fp"
 ↪About a minute ago   Up About a minute (healthy)
 ↪asterisk-ims
1a93b3ffe305   oaisoftwarealliance/oai-nrf:v1.5.1          "python3 /openair-nr…"
 ↪About a minute ago   Up About a minute (healthy)   80/tcp, 9090/tcp
 ↪oai-nrf
```

Chapter 5. Tutorials

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

Topic	Title
Overview	ARC-OTA Overview (October 2023)
Setup	ARC-OTA with SMC GH200: AI-Native Wireless Testbed (December 2024)
	ARC-OTA Development Environment Setup
Deployment	OpenAirInterface (OAI) 5G Core Network Deployment
	OpenAirInterface 5G Core Advance Deployment Using Docker-Compose

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 ARC-OTA platform. One research topic is integrating an end-to-end (E2E) O-RAN E2 interface within the ARC-OTA 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:

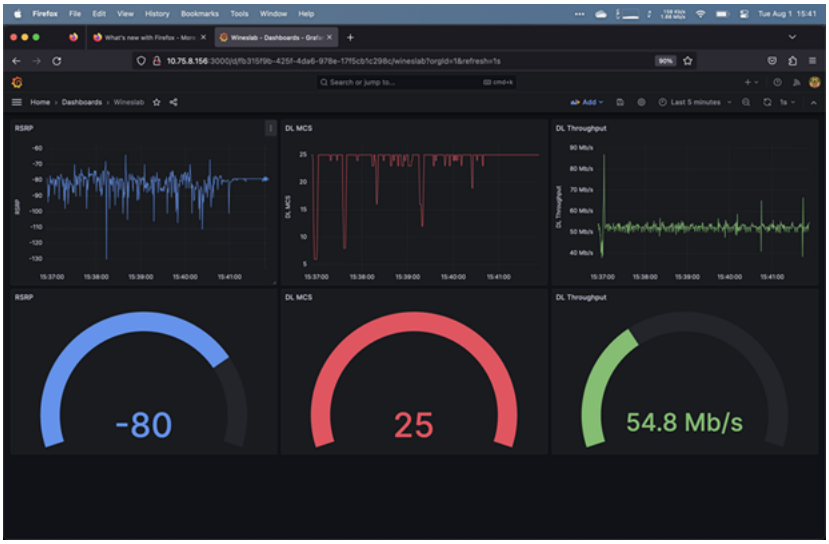
- A data-collection xApp running on an OSC RIC
- Deployed in a fully automated OpenShift cluster
- Connected to an InfluxDB database for telemetry storage
- Visualization of on a Grafana dashboard.

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 ARC-OTA 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 ARC-OTA 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: ARC-OTA 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 ARC-OTA platform, Open5Gs exhibited impressive performance:
  - **High Stability**: Maintained consistent operation during testing
  - **Sustained Performance**: Met the performance expectations set for the ARC-OTA 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 ARC-OTA 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

mps_sm_pbch: 8
```

(continues on next page)

(continued from previous page)

```
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 ARC-OTA 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 ARC-OTA extension and an opportunity to on-board ML-based applications for the developer community. Join ARC-OTA 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 NVIDIA Aerial RAN CoLab (ARC-OTA).	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 ARC-OTA. 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 ARC-OTA 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 ARC-OTA 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 ARC-OTA**: 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 RAN CoLab (Over the Air) 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 ARC-OTA 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 ARC-OTA 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)

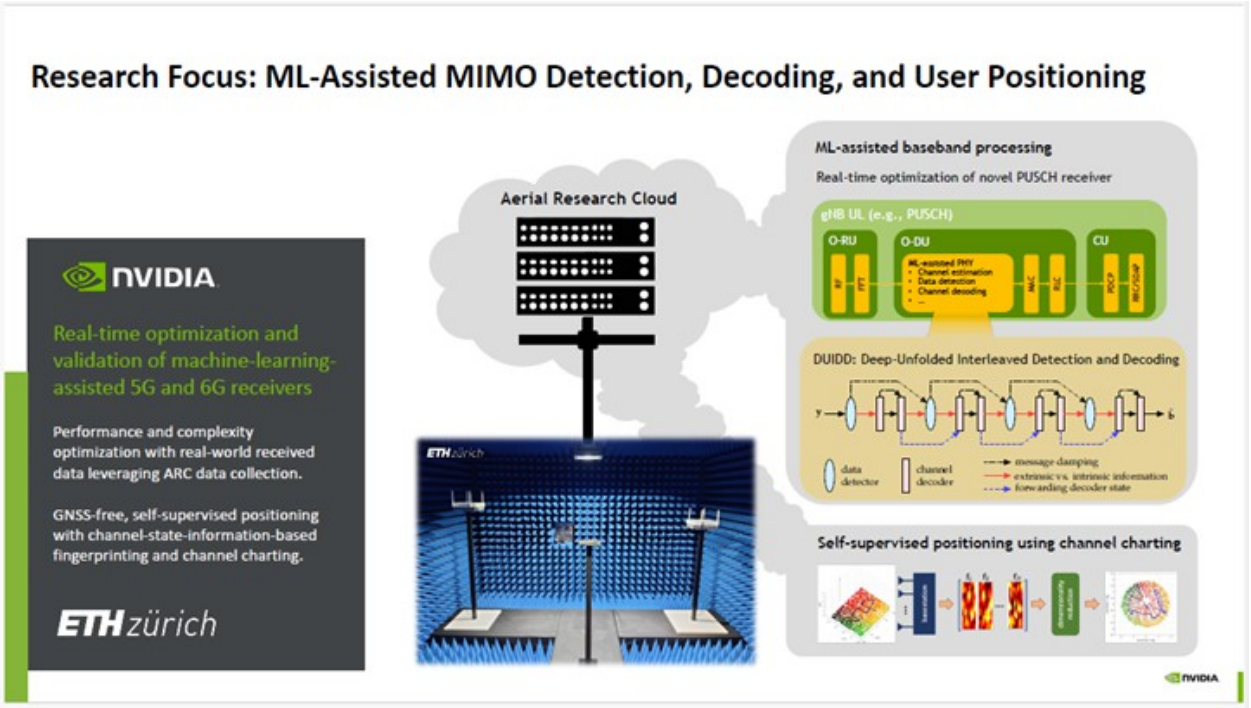
ML-Assisted Iterative MIMO Detection and Decoding

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 ARC-OTA, 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 ARC-OTA 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 ARC-OTA, integrating a GPU-based PHY layer.

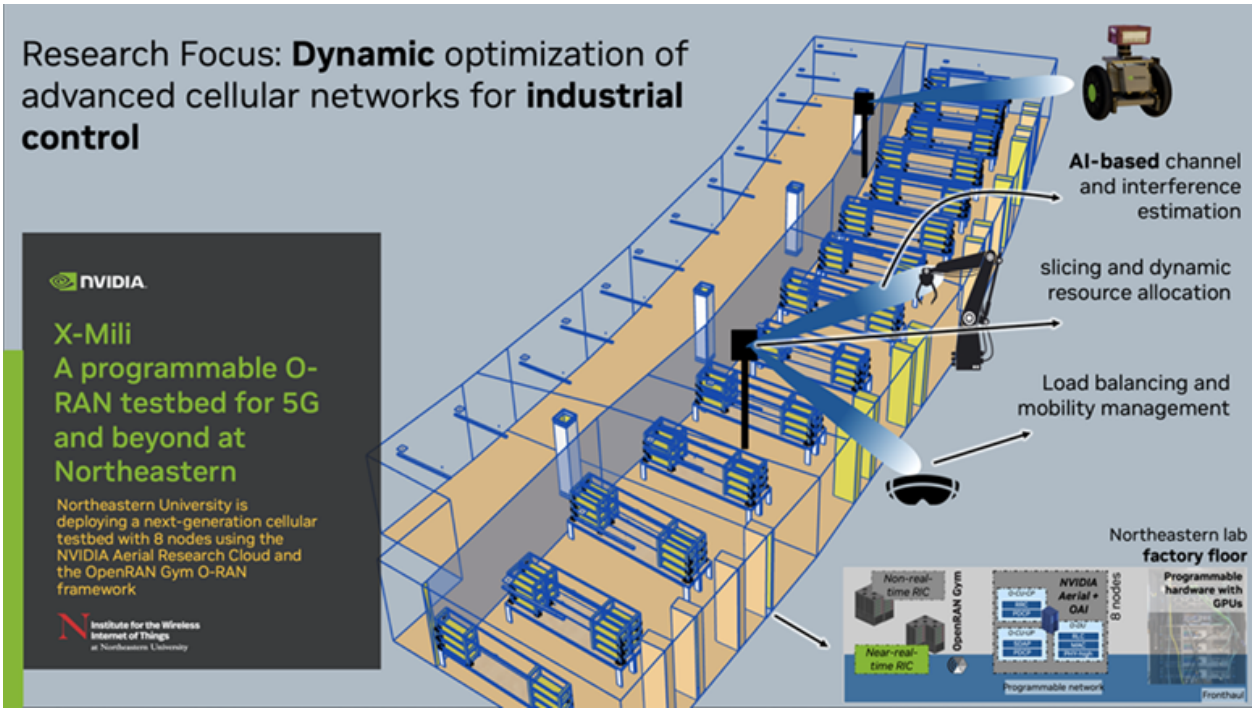
- The following UEs have been successfully tested in the system: OnePlus AC2003 Nord Samsung Galaxy S23, Sierra Wireless EM9191 NR 5G Modem, OAI Soft-UE.

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 ARC-OTA and OAI Demo Blog

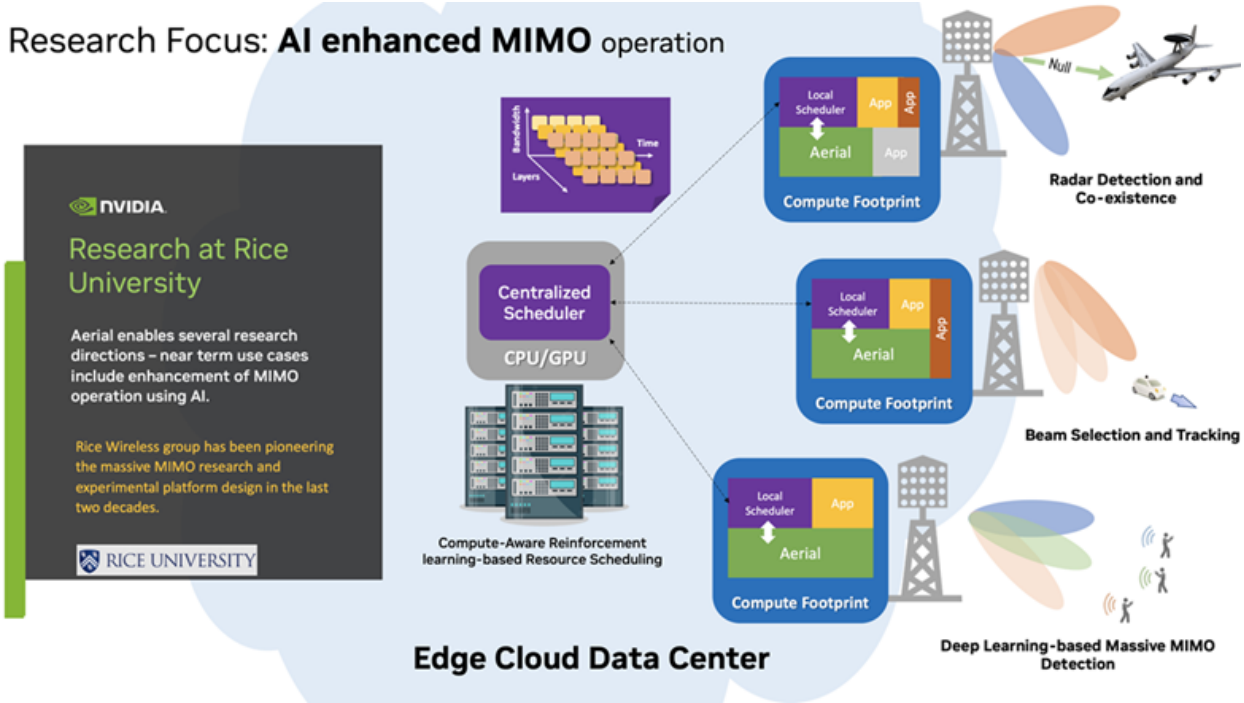
6.3.5. Rice University

Rice University outlines how NVIDIA ARC-OTA platform makes several key contributions to the research described in this blog post:

- ARC-OTA 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 ARC-OTA 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.

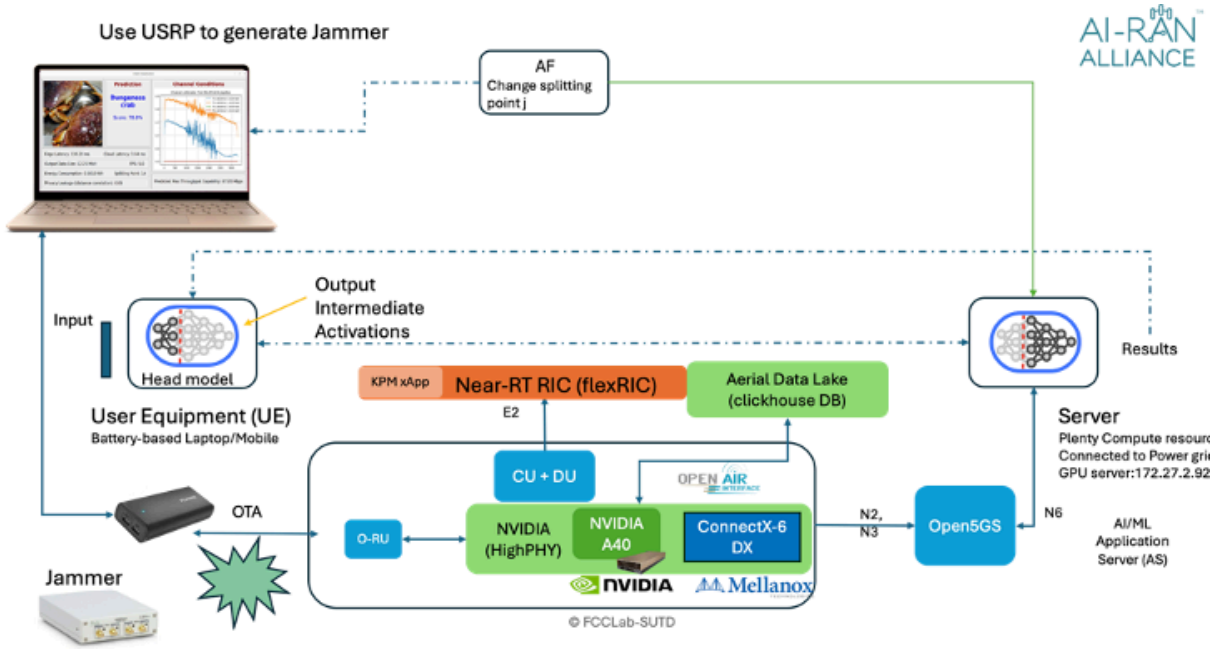
Visit https://wireless.rice.edu/ for information about the Rice Wireless program.



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 ARC-OTA 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.



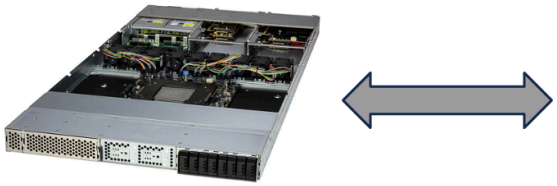
6.3.7. DeepSig Develops Algorithms for Learned Air Interface for 6G

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.

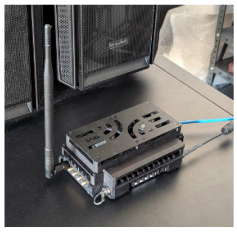
The hypothesis of this experiment is to challenge the current 5G air interface design that is model based with convenient assumptions on modulation, pilot, and frame design even though these are performance limiting. AI-native air interface allows AI to inherently design the waveform for a given site 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 ARC-OTA as the basis with a programmable UE implemented on a Jetson AGX Orin device:

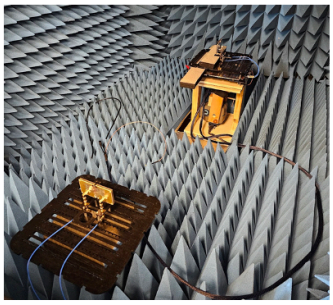
- Prototype 6G modem (OmniPHY) running on NVIDIA AI Aerial Platform which transceives information via neural encoding, receiving, and decoding
- Optimized to take advantage of GPU/DPU processing elements for low latency, high throughput, energy efficiency and for maximum capacity
- Can optionally perform 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 ARC-OTA 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 ARC-OTA 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 ARC-OTA BOM is located in the *Procure the Hardware* chapter of the Installation Guide.

Does the platform support MU-MIMO?

ARC-OTA 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?

ARC-OTA 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
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
```

(continues on next page)

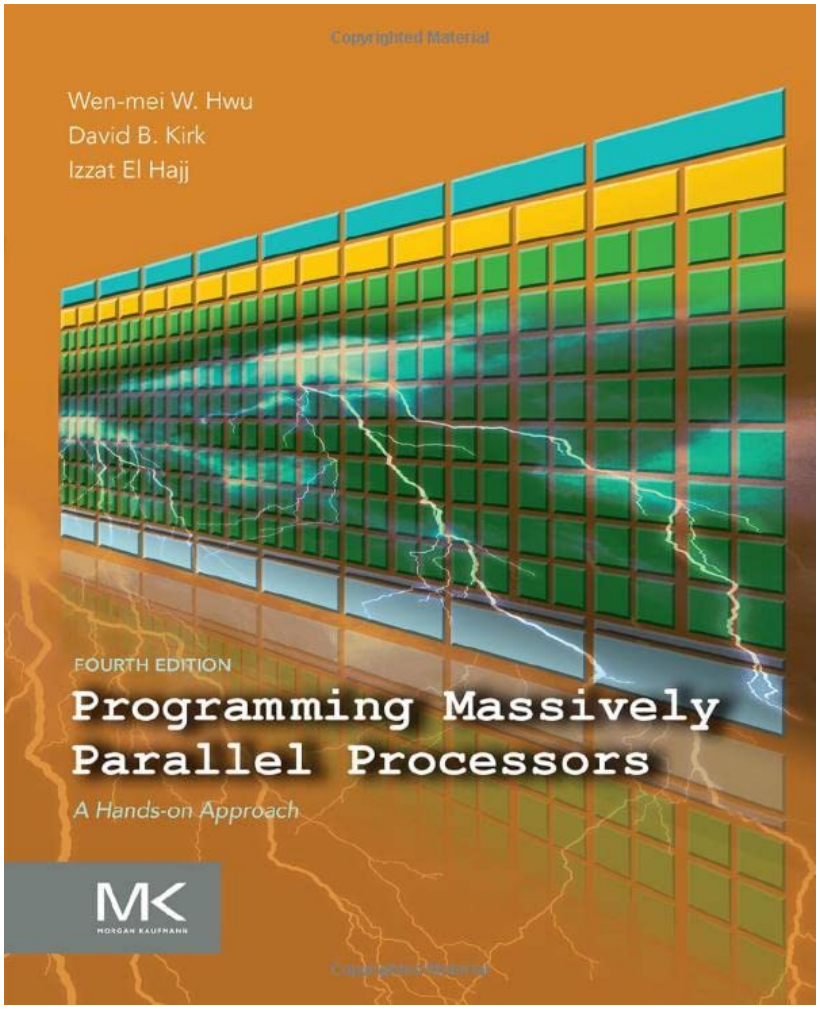
(continued from previous page)

```
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 Aerial Forum (6GDP Member Only Access)
- Contact us at aerial-info@nvidia.com

Chapter 8. Licensing

8.1. NVIDIA End User License Agreements

GOVERNING TERMS: The software and materials are governed by the NVIDIA Software License Agreement (found at https://www.nvidia.com/en-us/agreements/enterprise-software/nvidia-software-license-agreement/) and the Product-Specific Terms for NVIDIA AI Products (found at https://www.nvidia.com/en-us/agreements/enterprise-software/product-specific-terms-for-ai-products/).

8.2. OAI License Model

ARC-OTA 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. The OAI license model uses various licenses to distribute software and documentation and to accept contributions from individuals and corporations. The OAI Public License V1.1 provides the details for license itself.

Copyright