DEMO: Towards Fine-Grained and Automated Control for Large-Scale Wireless Digital Twins

Sarath Nagadevara
sxn0581@mavs.uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Afroze Rahman
afroze.rahman@uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Jiayi Meng
jiayi.meng@uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Abstract

The NVIDIA Aerial Omniverse^{TM} Digital Twin (AODT) is a state-of-the-art wireless digital twin (DT) platform for 5G and 6G R&D. The AODT provides intuitive graphical user interfaces (GUIs) that allow users to control a wireless DT, such as configuring radio settings and geospatial information of distributed units, radio units, and user equipments. However, these built-in GUIs impose significant constraints on fine-tuned control and automation, which are crucial for high-fidelity simulation of large-scale, real-world radio access networks. In this work, we propose an extension to the AODT that enables fine-grained control and automation without the need for GUIs. We demonstrate the use of this extension by running simulations within the AODT entirely through programmatic interfaces.

CCS Concepts

• **Networks** → **Network simulations**; **Mobile networks**; **Wireless access points, base stations and infrastructure**; **Programming interfaces**; *Network mobility*; *Network dynamics*; • **Computing methodologies** → **Simulation tools**.

Keywords

Wireless Networks, Digital Twins, Large-Scale Simulation, Programmatic Interfaces

ACM Reference Format:
Sarath Nagadevara, Afroze Rahman, and Jiayi Meng. 2025. DEMO: Towards Fine-Grained and Automated Control for Large-Scale Wireless Digital Twins. In *ACM SIGCOMM 2025 Conference (SIGCOMM Posters and Demos '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3744969.3748451

1 Introduction

A wireless digital twin (DT) is a high-fidelity virtual representation of a physical wireless network [2]. It accurately captures and simulates the network's intricate real-world attributes and interactions between its components—such as user equipments (UEs) and base stations—under different operational environments. It facilitates precise network planning and real-time, automated network management, e.g., by conducting what-if analyses across different network configurations. Its faster-than-real-time simulations also enable the collection of realistic, large-scale data of radio access networks (RANs) across diverse radio environments, which is essential for training accurate and generalizable machine learning (ML) models [1, 7].

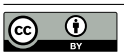
The NVIDIA Aerial Omniverse^{TM} Digital Twin (AODT) is a cutting-edge wireless DT platform, which supports the research and development of large-scale RANs for 5G and beyond [4]. It can simulate the physical and MAC layers of RANs within a realistic city-scale environment. The AODT provides graphical user interfaces (GUIs). Using these GUIs, users can manually deploy base stations and UEs within a wireless DT and specify their configurations, such as antenna panel settings and UE movement paths and speeds. Alternatively, users can define a zone where a specific number of UEs are procedurally spawned based on simple, pre-defined rules, e.g., allowed UE locations in the DT, using the GUIs.

While deploying and configuring base stations and UEs through the GUIs is intuitive, it fails to provide fine-grained control and automation—especially for large-scale simulations. In the AODT, users need to individually place each component within the wireless DT and configure their detailed attributes through point-and-click and drag-and-drop interactions. This can become error-prone and time-consuming. The alternative procedural spawning option for UEs can lead to non-deterministic or unrepresentative UE distributions if not properly constrained, which can potentially lead to inaccurate or misleading simulation outcomes. As a result, the automation capability of the AODT is limited by the lack of support for large-scale UE and RAN deployments and their fine-grained configurations. Without sufficient customization and automation, it is challenging to efficiently generate extensive, consistent simulation scenarios for comprehensive testing of next-generation RANs and rapid data generation for training ML models, etc.

To address these limitations, we extend the AODT framework by developing a configuration handler module, referred to as `ConfigHandler`. This module is a lightweight, easy-to-integrate plugin to the ADOT. It allows users to specify the properties of simulated distributed units (DUs) and radio units (RUs) of base stations along with UEs using programmatic interfaces *without* relying on the GUIs. To make it user-friendly, we also define a JSON schema that aligns with the properties supported by the built-in GUIs of the AODT. In this demo, we demonstrate how to leverage our developed `ConfigHandler` to configure UEs and RANs—including positions of DUs and RUs, UE movement paths and speeds, and antenna arrays to be used by RUs and UEs—across different simulated environments and perform simulations within the AODT.

2 Design of `ConfigHandler`

The proposed extension, `ConfigHandler`, comprises two main components. ❶ The first component is responsible for loading various



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGCOMM Posters and Demos '25, Coimbra, Portugal
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2026-0/2025/09
https://doi.org/10.1145/3744969.3748451

Table 1: Key-value pairs defined in the JSON schema for properties of DUs, RUs, and UEs in the AODT.

DU	
"id": int	"num_antennas": int
"fft_size": int	"subcarrier_spacing": float
"position": ("x": float, "y": float, "z": float)	"max_channel_bandwidth": float
RU	
"du_id": int	"panel_type": string
"cell_id": int	"du_manual_assign": bool
"height": float	"mech_tilt": float
"position": ("x": float, "y": float, "z": float)	"mech_azimuth": float
	"radiated_power": float
UE	
"user_id": int	"panel_type": string
"manual": bool	"waypoint_config": string
"is_indoor": bool	"waypoints": [{ "point": ("x": float,
"mech_tilt": float	"y": float, "z": float),
"radiated_power": float	"stop_sec": float,
"position": (x": float, "y": float, "z": float")	"speed_mps": float,
	"azimuth_offset_rad": float }]

properties of DUs, RUs, and UEs—including geospatial and radio attributes—from a user-defined JSON file through an implemented function, called `loadFromFile()`. ❷ The second component injects the loaded configuration data into an OpenUSD (Universal Scene Description) stage. OpenUSD is an open-source software platform that the AODT uses to create, edit, render, and simulate a 3D DT environment (referred to as *scene* in the AODT) [5, 6]. An OpenUSD stage is an instance of a USD data model that stores the data for simulations and serves as the entry point for accessing and manipulating the USD data used for simulations [3]. The implemented function for the second component is `addStageParams()`.

JSON Schema. To ensure that the input configurations comply with the AODT, we define a JSON schema. Table 1 highlights the primary key-value pairs defined in the schema for specifying the attributes of DUs, RUs, and UEs. In particular, users can define a UE movement path by specifying a sequence of their waypoints with corresponding speeds (unit: m/s), "`speed_mps`", and stop durations (unit: seconds), "`stop_sec`", in one JSON configuration file as the input to `ConfigHandler`. Note that the AODT can automatically adjust invalid waypoints, e.g., non-reachable locations, to the nearest reachable coordinates.

Integration into AODT. To integrate `ConfigHandler` into the AODT to run simulations without the GUIs, we invoke the two components of `ConfigHandler`, whenever a scene is opened. Opening a scene involves loading all scene settings, including its 3D environment and properties of DUs, RUs, and UEs, into the simulation engine of the AODT. To increase flexibility, we plan to further extend the AODT to support editing of an already opened scene. This will allow users to directly add, update, and remove DUs, RUs, and UEs and their properties within an existing DT in the AODT.

3 Execution Flow of AODT + `ConfigHandler`

In this section, we discuss how to run simulations in the AODT using the developed `ConfigHandler` without relying on the GUIs.

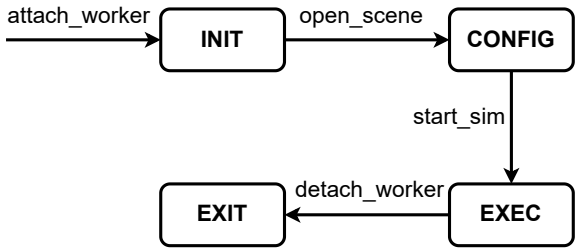


Figure 1: State machine of the AODT backend.

Table 2: State transition requests of the AODT backend.

Request	Description
`attach_worker`	Initialize a simulation worker in the AODT backend
`open_scene`	Configure the AODT backend with an OpenUSD stage
`start_sim`	Start a simulation in the AODT backend
`detach_worker`	Detach a worker from the AODT backend

To operate the AODT without the GUIs, we observe that users must follow the event sequence defined by AODT's state machine. This state machine tracks the state of a simulation. The simulation states include initialization (`INIT`), configuration (`CONFIG`), execution (`EXEC`), and exit (`EXIT`), as shown in Figure 1. With the GUIs, the AODT manages the data and control flow between the frontend GUIs and the AODT backend (which handles data processing/storage and simulations) through a set of backend handlers. These handlers process and respond to requests from the frontend. The simulation state is updated in the backend based on these frontend-driven requests, including attach_worker_request (`attach_worker`), open_scene_request (`open_scene`), start_ sim_request (`start_sim`), and detach_worker_ request (`detach_worker`), as illustrated in Table 2.

The execution flow without using the GUIs is as follows. At the start of each simulation, users should begin with an `attach_worker` request to transition the backend into the `INIT` state. Subsequently, an `open_scene` request should be sent to the backend to configure the simulation scene, during which `ConfigHandler` is executed. This triggers the backend to transition into the `CONFIG` state. Once the backend is in the `CONFIG` state, users can start the simulation using pre-defined DUs, RUs, and UEs, by sending a `start_sim` request to the backend to transition it into the `EXEC` state. Users can send a `detach_worker` request to detach from the backend, sending it into the `EXIT` state.

4 Demonstration

We demonstrate `ConfigHandler`, our extension to the AODT, on two testbeds—(1) a server running Ubuntu 22.02 on AMD Ryzen Threadripper PRO 5995WX 64-Cores Processor, with 67GB RAM, and 2 x NVIDIA RTX 4090 40GB GPUs; and (2) a server running Ubuntu 22.02 on AMD EPYC 7763 64-Core Processor, with 200GB RAM, and 2 x NVIDIA A100 80GB GPUs. We install the AODT on the servers, and run simulations with varying number of DUs, RUs, and UEs on an OpenUSD stage of Tokyo provided with the AODT, following the execution flow discussed in Section 3.

References

[1] Jakob Hoydis, Faycal Ait Aoudia, Sebastian Cammerer, Merlin Nimier-David, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. 2023. Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling. In *2023 IEEE Globecom Workshops (GC Wkshps)*. 317–321. doi:10.1109/GCWkshps58843.2023.10465179

[2] Latif U Khan, Ibrar Yaqoob, Muhammad Imran, Zhu Han, and Choong Seon Hong. 2020. 6G wireless systems: A vision, architectural elements, and future directions. *IEEE access* 8 (2020), 147029–147044.

[3] NVIDIA. 2024. Aerial Omniverse Digital Twin Documentation. https://docs.omniverse.nvidia.com/usd/latest/learn-openusd/terms/stage.html

[4] NVIDIA. 2025. *Aerial Omniverse Digital Twin.* https://developer.nvidia.com/aerial-omniverse-digital-twin

[5] NVIDIA. 2025. Scene Importer — Aerial Omniverse Digital Twin. https://docs.nvidia.com/aerial/aerial-dt/text/scene_importer.html

[6] Pixar Animation Studios. 2021. Introduction to USD — Universal Scene Description 25.05 documentation. https://openusd.org/docs/index.html

[7] Paolo Testolina, Michele Polese, Pedram Johari, and Tommaso Melodia. 2024. Boston Twin: the Boston Digital Twin for Ray-Tracing in 6G Networks. In *Proc. of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. 441–447. doi:10.1145/3625468.3652190