RAN-Wiser™ Hardware Agnostic RAN Framework for Truly Open RAN

Published Mar 2021

Table of Contents

Introduction

The 4G networks saw the advent of OFDM as the base for Layer 1 (L1) networks and an all IP network at Layer 3. In 5G networks, the major disruption is "disaggregation" and "virtualization." These two concepts are increasingly making the telco networks look like IT cloud networks. All the scaling benefits accrued in an IT cloud network are being brought to the telco cloud simplifying deployment, provisioning, and automation with the application of AI/ML.

Virtualization and the shift towards software implementations of most telco network components have been following "Moore's law." All layers down to the L1 in 4G networks were implemented in software; however in 5G, these are running as Cloud-Native Network Functions (CNF) on Commercial Off The Shelf (COTS) hardware. L1 implementations, on the other hand, always have needed specialized hardware-based signal processing architectures due to the inherent nature of that workload.

In 5G networks, there have been attempts to "virtualize" the L1 layer by implementing it in software with specialized Instruction Set Architectures (ISA). Disaggregation has been achieved by splitting the L1 signal chain into the Remote Radio Unit (RU) and the Distributed Unit (DU) "horizontally" and is bringing about limited pooling gains. The Open RAN (Radio Access Network) and the ORAN initiatives have advanced this by advocating open interfaces between the disaggregated components of the RAN. This horizontal disaggregation advocacy has helped spring up a new ecosystem of RU vendors. According to the new research data[1], at least 69% of MNO's are considering supporting open RAN initiatives by the year 2024. By 2026, open RAN will account for 58% of total RAN Capex spending and about 65% of sites deployed (of all sizes).

However, a lot still needs to be done on the DU front. The disaggregation in the DU landscape has propelled the separation of the DU hardware and software. DU High PHY has adopted a software-defined approach executing on COTS hardware. This has resulted in a diverse ecosystem of DU Independent Software Vendors (ISV) vendors. But the Independent hardware vendor (IHV) ecosystem is still very niche and results in vendor lock-in.

Source: 1 Rethink research - Open RAN adoption patterns and forecast 2020-2026

The COTS DU hardware methodology is not necessarily open, making it very difficult for operators to multi-tenant DU hardware platforms. Unlike a typical Virtualized Network Function (VNF/CNF) in the L3 software or IT world, a DU CNF is very closely tied to the underlying hardware implementation resulting in potential "lock-ins" with both a DU hardware and software vendor. Because of such closed proprietary designs, it is very difficult to deploy a multi-vendor DU solution, which is not in line with the Open RAN initiative's purpose.

More than 40% of the respondents to the research report indicated that their main goal is to bring down the total cost of ownership of RAN infrastructure. The majority of the COTS-based DU hardware today is based on a hardware platform from a single vendor. However, this is likely to change as the new hardware architectures that are better optimized to run RAN workloads and reduce the total cost of ownership of RAN infrastructure become available from multiple vendors. This will force the current DU software vendors to maintain multiple software versions for multiple hardware platforms. The MNO's will have to plan on a migration path to newer DU hardware platforms in the future without major software updates or downtime to their existing network.

True interchange of a DU ISV with a DU IHV can be possible only when the L1 DU software is truly "portable."

In terms of sophistication of the software tools and methodology, L1 PHY software is in its infancy. Most L1 PHY software is implemented in C and assembly as there is a tight coupling to the hardware underneath to get the most in performance and latency. Designing L1 PHY code for portability across hardware skews is virtually impossible, at least in the foreseeable future, using the traditional techniques. It is simply impossible to build an ANSI C compliant (the usual gold standard in portable C software) L1 PHY.

So how does one get to this goal of portable RAN software?

1. **Move from procedural C/C++ to more expressive Domain-Specific Languages (DSL):** Adoption of new languages is the best and cleanest way to introduce portability. But it has a long cycle time and needs industry evangelization and adoption.

2. **Open Framework based "Vertical disaggregation":** This is an API and a methodology-based framework that has been adopted in other disciplines such as graphics and AI.

Saankhya Lab's RAN-Wiser™ platform attempts to solve this problem by taking the approach of vertical disaggregation. RAN-Wiser™ is a standardized software framework for RAN workloads which is integrated into a Telco cloud infrastructure software. The framework provides a uniform model for the hardware underneath and lowers the barrier to build portable RAN software.

The RAN-Wiser™ consists of a

1. A "static" methodology part that requires DU CNF (L1 software) to be developed using the APIs provided by the RAN-Wiser SDK.
2. A dynamic runtime system that comprises the Super-Scheduler and the Hardware Architecture Description.

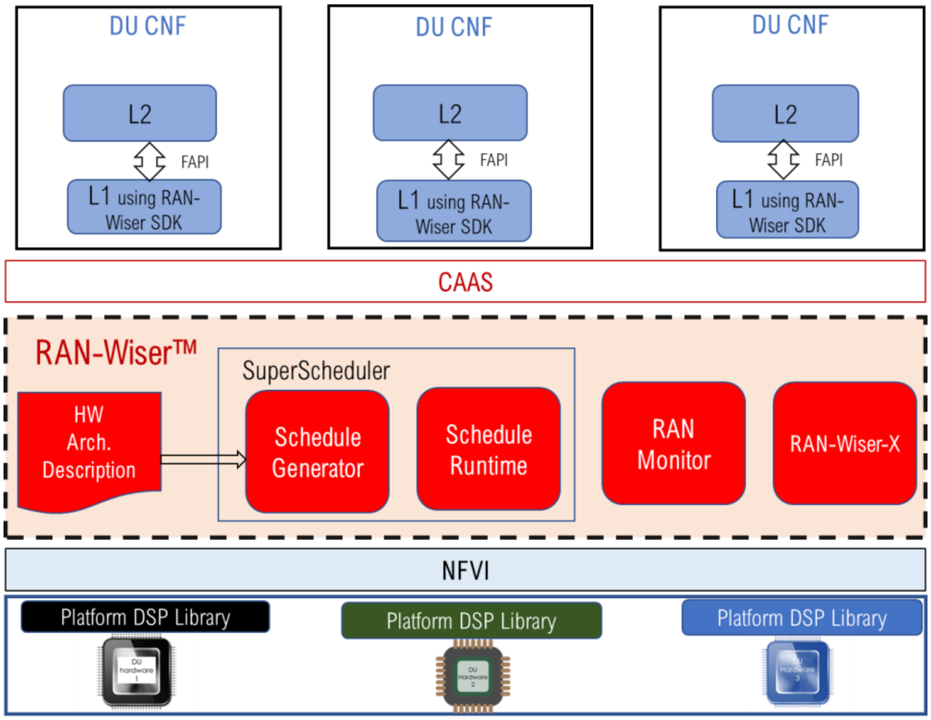
The SDK allows software developers to express RAN tasks (developed as C functions) as a pipeline or a Data Flow Graph (DFG), thus modelling the domain behavior. The run time system is responsible for scheduling execution of these tasks on the different kinds of processing elements (GPPs, GPUS, DSPs, fixed-function accelerators, etc.) of the underlying hardware platform. It is an infrastructure component executing on the server blade and thus acts as an additional indirection between the RAN hardware and the DU CNF. Unlike a traditional IT hypervisor, this runtime component imposes no overhead or latencies. It dis-aggregates and virtualizes the hardware from the RAN software.

The virtualization of the hardware is achieved through the hardware architecture description that is supplied with each hardware skew as a configuration file. Along with the DSP vector libraries, this abstracts the hardware and allows for mapping the L1 software to the underlying DU platform.

This whitepaper outlines the RAN-Wiser™ portable RAN development framework and its use cases.

RAN-Wiser™ Portable RAN Development framework

RAN-Wiser™ portable RAN development framework facilitates the development of Containerized RAN Network Function(CNF). The CNF developed using RAN-Wiser™ can be deployed on multiple DU platforms without any source code level changes. RAN-Wiser™ provides the telecom operators with the ability to “slide-in” any new DU hardware without any vendor lock-in.



The key elements of the framework are as given below:

- **Application or development components**
    - **RAN-Wiser™ RAN SDK and RAN-Wiser-x**: RAN-Wiser™ RAN SDK is the set of APIs used by L1 developers to develop the RAN tasks that constitute the CNF. RAN-Wiser-x is the tool that extracts the data flow graph (DFG) or the task graph of the CNF. This is used by the Super-Scheduler.

- **Infrastructure components**

- This comprises the Super scheduler and the Hardware Architecture description file used to schedule the DFG generated by RAN-Wiser-x on the target hardware platform.
- Platform Drivers required for communication with the DU accelerator platforms

RAN-Wiser™ RAN SDK

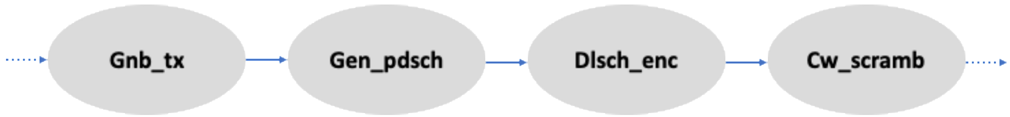
RAN-Wiser™ RAN SDK is a lightweight API set that enables L1 software developers to express a RAN CNF as a pipeline or data flow graph (DFG) of RAN tasks. Each task is developed as a C function with a "*call next*" to the next task in the pipeline, and each task uses *"wait_signal"* to block till data from the previous task is available.

RAN-Wiser-X

RAN-Wiser-X is the tool used by L1 developers for the following purposes:

- Compile-time checks to ensure the RAN-Wiser™ APIs are used correctly in the C code and there are no inconsistencies or errors.
- Extract the data flow graph (DFG) from the L1 code. The Super Scheduler will use this DFG
- Compile the L1 C code by invoking the target device compiler as a part of its execution.

An example of a DFG extracted by the RAN-Wiser-X is given below:



Super Scheduler

The Super Scheduler consists of two components:

- Schedule Generator
- RAN-Wiser™ runtime scheduling library

Schedule Generator

The Schedule Generator performs the job of assigning each node in the DFG generated by RAN-Wiser-x to one or multiple processing elements (processor cores, DSP cores, etc.) of the DU accelerator platform. To do this, Schedule Generator analyzes the Hardware Architecture Description of the DU platform and performs sophisticated graph partitioning to identify the best possible schedule for all the tasks in the DFG, i.e. it assigns a PE(s) to each of the tasks. The

Schedule Generator's output is stored in a file used by the RAN-Wiser™ runtime scheduling library.

The Schedule Generator can assign the RAN tasks in the DFG to the processing elements on a single chip in the DU platform or across multiple chips within the same DU platform or across multiple DU platforms. The Schedule Generator is a non-runtime tool and hence does not cause any runtime overheads.

RAN-Wiser™ runtime scheduling library

This library contains the runtime implementation of the RAN-Wiser™ RAN SDK APIs and the runtime scheduler implementation. This library is linked with the object files generated after the compilation of the L1 C-code to produce the final L1 implementation.

As a part of one-time initialization, this library reads the output generated by the Schedule Generator. It loads the RAN tasks on the identified processing element before starting the L1 execution from the first task in the DFG. Since the loading of these RAN tasks is done only once per RAN CNF instantiation and is performed before actual task execution starts, it does not result in any runtime overheads.

Benefits

The RAN-Wiser™ has other benefits than just portability. It introduces a design paradigm that allows a clean separation of concerns. Other major benefits include

1. Virtualization of the RAN resources: Fine grain control allows efficient implementation of RAN slicing functionality
2. RAN Monitor provides hyper Observability, enabling the development of AI/ML xApps for power, performance management, and network tuning.
3. Domain level thinking and problem abstraction
4. Security: Additional level of indirection allows security checks of the CNF so that there is no spoofing.
5. DU CNF functionality audit for Power, Performance, Security before deployment

Conclusion

This whitepaper describes the development framework for a portable RAN solution. With such a framework available, the operators are no longer tied to a single hardware vendor and can move across multiple hardware vendors. This solution will allow the mobile network operators to work with custom RAN hardware without sacrificing COTS solution benefits. This is an important step in the development of a Truly Open Virtualized RAN.