tsdsi
India's Telecom SDO

Telecommunications
Standards Development
Society, India

Feasibilty of **OPEN-SOURCE** for **5G**

JULY 2021

Table of Contents

About the Report

Towards the latter part of 2020, TSDSI commissioned a Task Force for investigating the feasibility of using Open-Source for the development of the network infrastructure functions aimed at 5G. The Task Force was established with a call for participation to all the TSDSI members and the extended community. The Task Force had an active participation from at least 30-40 subject matter experts belonging to over 15 separate organizations. It ran from late 2020 to July 2021, with broad participation from carriers, vendors and academia on a voluntary contribution basis and resulted in the conclusions and recommendations, which are summarized herein. The report summarizes the background for the study, provides a detailed account of each of the three areas of investigations and suggests a course of action for future work.

Summary Report

Background

1 - Transformative Nature of Open-Source

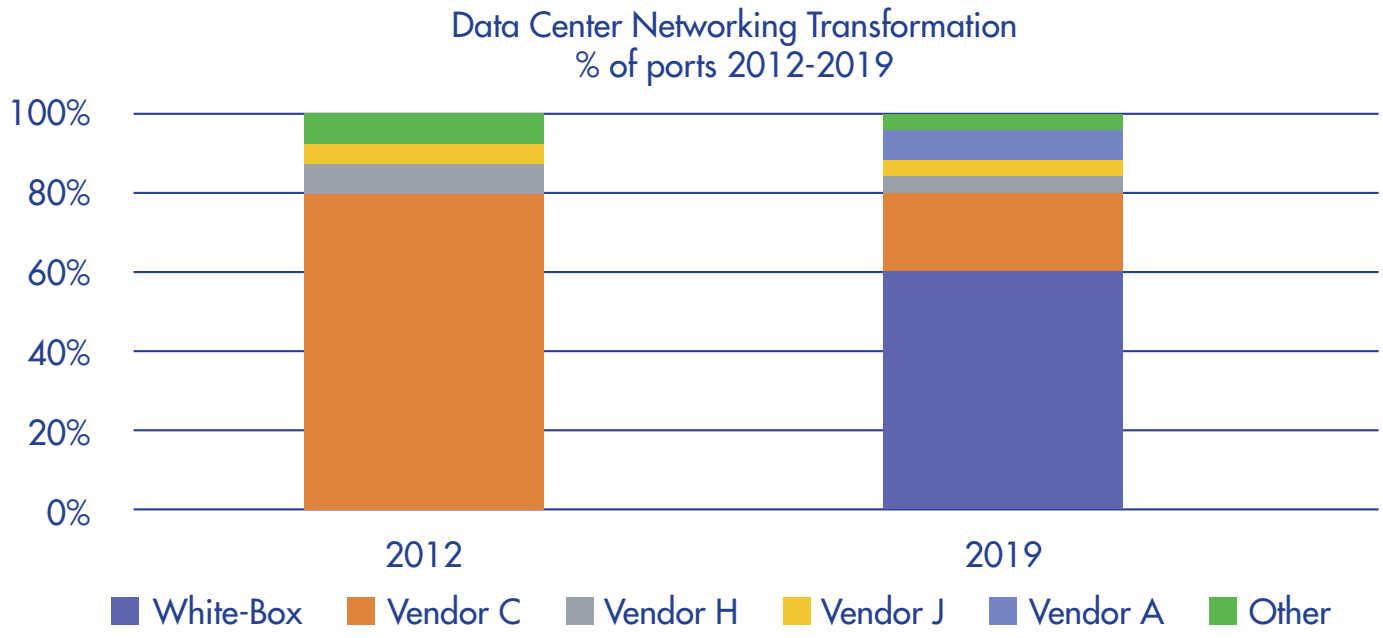


Figure 1: Data Centre Networking Transformation

In the past decade, Open-Source has transformed many industries and it seems that Telecom is in the midst of such a transformation partly due to the nature of technological changes with 5G and with the opening of a lot of interfaces that were otherwise proprietary to the incumbent vendor ecosystem. As an example, Data Centre Networking was totally changed in the period 2012 to 2019 as illustrated by the bar chart of Figure 1. In 2012, the industry was dominated by the proprietary closed systems, with a single dominant vendor. By 2019, the picture had changed completely, with White-Box solutions using disaggregated Open-Source software dominant. This rapid change was driven by several factors:

- The "Software-ization" of networking through NFV/SDN.
- The emergence and dominance of cloud service providers
- The viability of high quality Open-Source networking solutions from organization such as the Linux Foundation

Many of the same factors are at play in the Telecom space and it is possible that the transformation that occurred in Data Centre Networking will also happen in Telecom.

Several large carriers with global impact (e.g., AT&T and DT) have made significant bets on the emergence of Open-Source for 5G. AT&T has declared Open-Source a pre-requisite for procurement, though this is not applied in a hard and fast manner. The role of the Indian ecosystem and of Indian standards bodies such as TSDSI in the adoption of Open-Source in 5G is an important issue. Many experts have suggested that India can take leadership in this and show the way for the rest of the world.

2 - Open-Source is Linked to the Telecom Standardization Process

Common Objective with Different Methodology	
Reference Code	Document specification
Transparent & Collaborative implementation effort	Transparent & rigourous process
Faster iteration Phase 3	More deliberative Products

Figure 2: Relationship between Open-Source and Standards

There is a strong relationship between standards bodies and Open-Source efforts. Both have very similar objectives - namely to foster interoperability, increase industry cooperation where it can advance overall good and to get convergence on common frameworks that can provide a base for innovation. As shown in Figure 2, the approach to achieving these goals is sometimes different. However, in recent times we are starting to see a blend between standards and Open-Source as Telco standards bodies foster Open-Source activities and vice versa. An example is the launch of Open-Source MANO effort by ETSI as a way of providing a common framework for orchestrating and managing Telco services, where a traditional SDO has ventured into the running and Open-Source project. Similarly, many of the SDN efforts from the Linux Foundation (e.g., OpenDaylight) have driven standardization of the APIs used by network applications to control network resources.

This blending of standards and Open-Source has made it imperative that TSDSI actively engages in Open-Source exploration.

3 - Open-Source Benefits are Compelling

The move to Open-Source will create benefits across the entire Telecom ecosystem. Many of these benefits are already seen in other adjacent sectors.

- **Mobile Operators:** Carriers will see the benefit of faster innovation, vendor independence and lower capital costs. This has been borne out of some of the experiences of early adaptors such as AT&T. But it is clear that the path is far from easy, and these benefits are not always apparent at the start of the journey.

- **Vendors:** Vendors see the benefits of reduced development costs and reduced time-to-market. By building on a standard platform, they eliminate major developmental effort and reduce development risk. In addition, they increase the pool of available technical talent already familiar with their development environment.

- **Start-up Companies:** Start-ups will benefit the most of all. Eliminating risk as well as reducing development costs and time are critical for the emergence of new companies. In addition, start-ups can focus on the innovative value they can bring, rather than recreating the plumbing underneath to support generic functionality.

- **Government:** Governments will see the emergence of a more robust, indigenous 5G vendor eco-system. This creates the opportunity for independence from foreign suppliers. In addition, the transparency of the Open-Source development process and its availability for inspection, ensures the absence of back-doors and other potential security exposures. This is critical for infrastructure of national importance.

4 - But Open-Source Is Confusing and Needs Curation

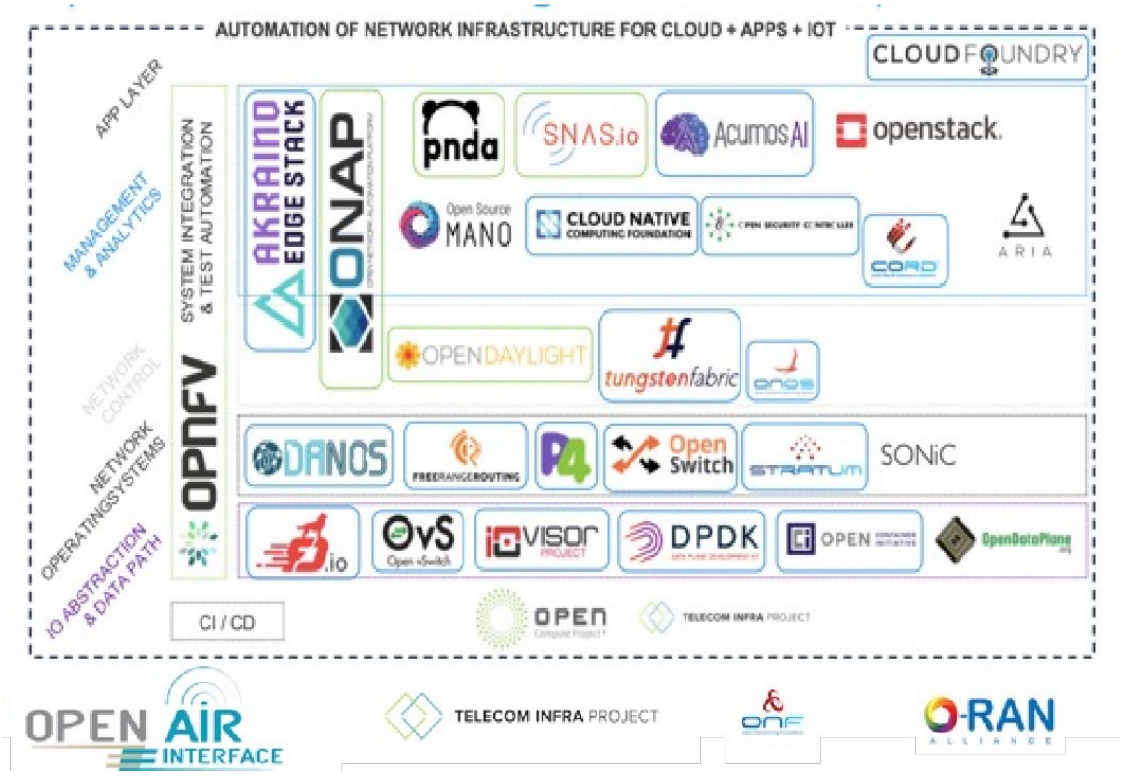


Figure 3: Proliferation of Open-Source Networking and Telecom Projects

There are numerous Open-Source projects that have been initiated in the past few years in the Telecom and Networking space (as shown in Figure 3). These include projects initiated by established Open-Source groups such as the Linux Foundation; projects from more specialized Open-Source bodies such as Open Networking Foundation; projects initiated by vendor communities such as Cloud Native Computing Forum; projects initiated by Cloud service providers such as Telecom Infra Project; and projects initiated by the carriers such as ORAN and OPNFV.

The proliferation of Open-Source components in the Telecom space is a mixed blessing. On the one hand, it provides many choices of components. On the other hand, many of these choices are overlapping and confusing. To build a coherent end-to-end system requires considerable care in component selection and some effort in integration and testing. In cooking terms, there are many "ingredients" available, but to make a complete dish we need a good "recipe".

About the Task Force

1 - Objectives of the Task Force

The TSDSI Task Force was commissioned to study Open-Source for 5G. The questions considered by the Task Force included some of those mentioned below:

- Which Open-Source components or project are most pertinent to 5G?
- Can components of the 5G stack be entirely built from Open-Source? Which ones and how?
- Can we provide an Open-Source "recipe" to assist and accelerate Indian industry participation in the 5G ecosystem?
- Can Open-Source level the playing field for Indian 5G product companies?
- What specific actions need to be taken? Standards? IPR protection? Open-Source Industry Consortia? Policy advocacy?

Summarized more succinctly, the specific goals of the task force were:

- Study and evaluate Open-Source components needed for an end-to-end 5G system.
  - » Key metrics: Quality, Maturity, Completeness, Usage, etc.
- Create a study report to document the "recipes" for creating Open-Source 5G.
  - » Recommend the best available components and define the gaps to be filled and integration efforts needed.
  - » Define some pilot projects to demonstrate the effectiveness of the recommendations.
- Make recommendations on how to move forward with the following questions in mind
  - » How should Open-Source interact with the Telecom standardization process in collaboration with industry?
  - » How can Open-Source assist in accelerating the Indian 5G vendor ecosystem?

2 - Structure and Operation of the Task Force

The Task Force was constituted and operated under the aegis of the Standardization Committee of TSDSI. The members of Task Force were recruited on voluntary basis through an open call for participation.

The governance of the Task Force was jointly handled by two of participating leaders from industry, who in turn reported to the Standardization Committee.

Each sub-team met once a week for the technical discussions and reviews and made progress over the duration of the Task Force. In addition, there were bi-weekly meetings of supervisory group to discuss issues, resolve differences and take important decisions with regard to the direction of each sub-team and their work streams. The supervisory group periodically met with the Standardization Committee to report progress as well as seek feedback and direction on the work efforts.

End-to-end 5G deployment scenarios were studied to identify key areas where availability of Open-Source components could give a major boost to the efforts of creating complete products. These components should be the "plumbing" needed to build more functions and specific implementations needed by the industry in general. Given the diverse technology expertise and the vastness of the solution space, the Task Force was divided into three sub- teams to address each of the Radio Access, IP transport and Applications, sub-domains of 5G.

The leaders of each sub-team are mentioned below. The overall participant list is much larger and is provided in Addendum 1. TSDSI is truly grateful to these volunteer experts who made this effort possible and devoted significant time to this activity.

Radio Access sub-team leaders:

- Ambarish A (Sookhta)
- Ankur Chauhan (Airtel)

IP Transport sub-team leaders:

- Abhijit Chaudhary (Niral Networks)
- Harpreet Kaur (Hughes Systique)

Applications sub-team leader:

- Prabhu Rajashekaraiah (Airtel)

Summary of Objectives and Results of Sub Teams

In order to drive the activities with specific goals and directions, each sub-team debated and identified one or more problem statements and defined the same to the level of detail that is needed if a product were to be built solving that specific problem statement. This allowed the sub-teams to evaluate the Open-Source components available against a set of objectives and helped them identify the gaps and the work to be doneto make the Open-Source components usable in a real-life environment. Here is a summary of key problem statements and the outcomes of the study undertaken by each team:

A. **Radio Access** - This sub-team looked at the 3GPP reference architecture for 5G Radio access network and chose to investigate the feasibility of using Open-Source components for implementing CU and DU. This team evaluated OpenAirInterface (OAI) and ORAN Software Community (ORAN-SC) projects using a set of criteria defined to test the viability of building a commercial grade software solution for CU and DU. This team concluded that OAI was better placed amongst the two Open-Source projects. The detailed reasoning and evaluation parameters are part of the detailed report from that sub-team (Addendum 2).

B. **IP Transport** - The Transport sub-group worked on the analysis and evaluation of the Open Source software and open hardware design for Packet Transport Network of the 5G. The packet transport network is used to forward 5G traffic over an IP/MPLS backbone to the 5G Packet Core. The transport sub-group mainly focused its study on 3 components:

   i. Disaggregated Cell Site Gateway (DCSG)
   ii. Edge Router (ER)
   iii. Software Defined Networking (SDN) Controller

For DCSG and ER, the sub-team evaluated Open-Source components such as DANOS, FRRouting and for SDN controller ONOS and ODL were investigated against a number of criteria defined to test the maturity and readiness of the Open-Source projects. The final recommendations were to use a combination of DANOS and FRRouting for DCSG and ER, whereas ODL edged ahead of ONOS for SDN controller implementation. The details of these recommendations are available in the detailed report from this sub-team (Addendum 3).

C. **Applications** - The focus of the Applications sub-team is on the RIC framework as defined by 3GPP and ORAN reference architecture. RIC framework will provide the platform to host the APPs to optimize the RAN functionalities like RRM, SON etc. RIC framework will also provide the open APIs for the APP development. RAN Intelligent Controller (RIC) platform leverages "compute" capabilities of a cloud-native environment to enable AI/ML driven intelligent decisions and automation in the RAN. The Non-RT control functionality (> 1s) and near- Real Time (near-RT) control functions (< 1s) are decoupled in the RIC. Non-RT functions include service, configuration and policy management, RAN analytics and model-training for the near- RT RAN functionality. rApps are applications designed to run on the non-RT RIC whereas the Near-RT RIC hosts xApps to provide value added services for near RT functions in RAN.

The sub-team concluded that for Near-RT RIC, recommendation is to use SW from SD-RAN. But for Non-RT RIC, O-RAN provided solution can be used, as currently it is the only available solution in the Open-Source domain. The details of these recommendations are available in the detailed report from this sub-team (Addendum 4).

Next Steps and Recommendations

The Task Force has conducted extensive paper study of the Open-Source components and the feasibility of their usage in solving some of the problem statements pertinent to 5G network implementations. There has not been any effort to test the premises and conclusions, as that was not an objective of this task force. This can be taken as Phase 1 of an effort to propagate Open-Source within the Indian 5G ecosystem.

Therefore, a logical next step could be to undertake implementation of some of the suggested pilots in this report. This would be Phase 2 of the activity. The Task Force feels that a Phase 2 is a necessary step without which the value of the work would not be realized. Some observations on the structure and value of Phase 2 are mentioned below:

- The Phase 2 pilot implementations suggested in the report can be modified to orient towards testing the 5Gi standard implementations. This would be a good objective for the next phase of TSDSI's Open-Source exploratory work.

- Phase 2 practical implementation should be undertaken as downstream projects of the recommended Open-Source projects. Most of the work will be in integration and testing, with a small amount of new code.

- The Phase 2 work cannot be done as a volunteer effort. To be successful, it will require dedicated funding and a professional staff of experienced software developers. This is the only way to get right software development expertise and target suitable hardware/ firmware and computing platforms to demonstrate the implementations

- The professional team can be hosted in one or more academic institutions who can then get funded by government and industry. An academic setting will maximize the opportunities of governmental funding and will also ensure the independent and technically unbiased nature of the work. An example of a successful similar model is the Open Network Foundation, based in Stanford University.

- Industry participants (and participants from academia and other sectors) would be encouraged to get involved. The activity would be structured as a well-governed Open-Source Consortium, operating as a meritocracy without any proprietary or parochial bias. In fact, industry participants would be encouraged to take ownership of certain components, with full committer rights for qualified individuals.

Phase 2 can create strong high-quality reference implementations for key components within the 5G system. Commercial products for the 5G space could readily follow Phase 2 and be based wholly or partly on the reference implementations of Phase 2. We expect the Indian vendor ecosystem to rapidly benefit.

The phased approach is summarized in Figure 4 below.

	Objective	Structure	Value	Status
Phase 1	Feasibility	TSDSI task force with broad industry participation: Voluntary	Guideline for future efforts.	Completed.
Phase 2	Reference implementation and pilots	TSDSI Role? Industry involvemnet, Academic involvement, Needs funding	5G open source reference to accelerate development for Indian (and Global) companies	To be done: Initial exploration with academia shows strong interest.
Phase 3	Products	Private company led. Business driven	New Indian (and Global) companies participates in 5G	Future

Task Force Participants

TSDSI is immensely grateful to the following volunteers who have generously given their time, energy and expertise to this effort. TSDSI is also grateful to their employers who supported their participation in this activity.

1 - Overall Task Force:

Leaders	
Manish Gangey	Airtel
Inder Gopal	IISc

Participants	
Abhijit Chaudhary	Airtel
Abhishek Bhalotia	IISc
Ambarish	Sooktha
Amit Gupta	TSDSI
Anand Kumar	STL
Anindya Saha	Saankhya Labs
Ankit Tripathi	STL
Ankur Chauhan	Airtel
Ashwani Kumar	Huawei
Pamela Kumar	TSDSI
Dhiraj Talele	STL
Dhruv Dhody	Huawei
Guna B Shekar	TSDSI
Guru Shankar	Saankhya Labs
Gururaj	Saankhya Labs
Harpreet Kaur	Hughes Systique
Hemant Rath	TCS
R Prakash	CDOT
Sandeep Agrawal	TCS
Sreenath	Lekha Wireless
Srinivasan Radhakrishnan	Sooktha
Indranil Basu	STL
Jacob Mathew	Wipro
Jishnu	Tejas Networks
Kranthi Molleti	STL
Leena Ambawani	CDOT

Madhu Sudan	IIT-M
Manju	Lekha Wireless
Minu Singh	TCS
Munish Bhardwaj	Airtel
Niranth Amogh	Huawei
Naveen Arora	TCS
Nitish	Qualcomm
Nicolas	Qualcomm
Powen	Edge Core
Prabhu Rajashekaraiah	Airtel
Raj Patel	Edge Core
Rajesh Mhatre	STL
Ramu	Lekha Wireless
Ravi Lakhotia	Vodafone Idea
R Prakash	CDOT
Sandeep Agrawal	TCS
Sreenath	Lekha Wireless
Srinivasan Radhakrishnan	Sooktha
Sudarshan	Saankhya Labs
Swaminathan Seetharaman	Wipro
Sujay Gupta	IP Infusion
Vivek Sreevatsan	Cisco

2 - Radio Access Sub-Team:

Leaders	
Ankur Chauhan	Airtel
Ambarish	Sooktha

Participants	
Anindya Saha	Saankhya Labs
Pamela Kumar	TSDSI
Guna Shekar	TSDSI
Gururaj	Saankhya Labs
Jishnu	Tejas Networks
Kranthi Molleti	STL
Madhu Sudan	IIT-M
Niranth Amogh	Huawei
Nitinsh	Qualcomm
Nicolas	Qualcomm
Ramu	Lekha Wireless
Ravi Lakhotia	Vodafone Idea
R Prakash	CDOT
Sreenath	Lekha Wireless
Srinivasan Radhakrishnan	Sooktha
Sujay Gupta	IP Infusion
Vivek Sreevatsan	Cisco

3 - Transport Sub-Team:

Leaders	
Abhijit Chaudhary	Niral Networks
Harpreet Kaur	Hughes Systique

Participants	
Abhishek Bhalotia	Airtel
Ashwani Kumar	Huawei
Pamela Kumar	TSDSI
Dhruv Dhody	Huawei
Guna B Shekar	TSDSI
Guru Shankar	Saankhya Labs
Jishnu	Tejas Networks
Krishna Kumar	STC
Jacob Mathew	Wipro
Jishnu	Tejas Networks
Kranthi Molleti	STL
Leena Ambawani	CDOT
Madhu Sudan	IIT-M
Manju	Lekha Wireless
Minu Singh	TCS
Munish Bhardwaj	Airtel
Niranth Amogh	Huawei
Naveen Arora	TCS
Nitish	Qualcomm
Nicolas	Qualcomm
Powen	Edge Core
Prabhu Rajashekaraiah	Airtel
Raj Patel	Edge Core
Rajesh Mhatre	STL
Ramu	Lekha Wireless
Ravi Lakhotia	Vodafone Idea
R Prakash	CDOT
Sandeep Agrawal	TCS
Sreenath	Lekha Wireless
Srinivasan Radhakrishnan	Sooktha

Sudarshan	Saankhya Labs
Swaminathan Seetharaman	Wipro
Sujay Gupta	IP Infusion
Vivek Sreevatsan	Cisco

4 - Application Sub-Team:

Leaders	
Prabhu Rajashekaraiah	Airtel

Participants	
Abhijit Chaudhary	Niral Networks
Ambarish	Sooktha
Amit Gupta	TSDSI
Anand Kumar	STL
Ankit Tripathi	STL
Ankur Chauhan	Airtel
Ashwani Kumar	Huawei
Pamela Kumar	TSDSI
Dhiraj Talele	STL
Guna B Shekar	TSDSI
Guru Shankar	Saankhya Labs
Gururaj	Saankhya Labs
Harpreet Kaur	Hughes Systique
Hemant Rath	TCS
Inder Gopal	IISc
Kranthi Molleti	STL
Leena Ambawani	CDOT
Madhu Sudan	IIT-M
Manish Gangey	Airtel
Minu Singh	TCS
Munish Bhardwaj	Airtel
Niranth Amogh	Huawei
Naveen Arora	TCS
Nitish	Qualcomm
Nicolas	Qualcomm
Rajesh Mhatre	STL
Ravi Lakhotia	Vodafone Idea
Sandeep Agrawal	TCS
Swaminathan Seetharaman	Wipro
Sujay Gupta	IP Infusion
Vivek Sreevatsan	Cisco

Findings of Sub Task Force

Radio Access Network Sub-Team

1 - Introduction

This addendum is the report of the Sub-Team on Radio Access Networks within that Task Force and supports the "Feasibility of Open-Source for 5G- Final Overall Report" issued by the TSDSI Task-Force on Open-Source for 5G. Please read the main section of the report to get the context, background and overall recommendations of the Task Force.

1.1 Scope

This addendum captures the feasibility of Open-Source usage in the 5G Radio Access Network. Below sections describe the scope of the work done and how the document is further organized to detail out on activities to be taken as part of the short-term goal for the group. Short term goal includes below:

- Identify the Open-Source ecosystem components in 5G RAN.
- Identify the gaps in Open-Source components for their adherence to standards.

In the current study, the focus is on the CU and DU as depicted in the below diagram.

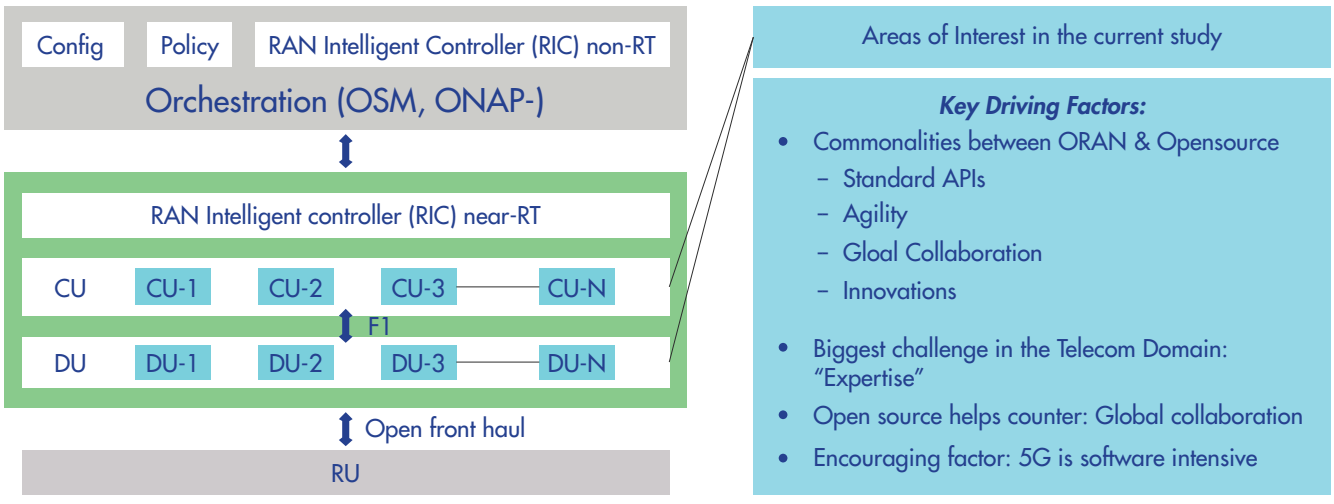


Figure 1: Scope of Task Force Study

Deployments in 5G as defined by the 3GPP are as depicted below. Immediate focus on a lot of deployments in the existing infrastructure is based on the non-Standalone and green-field deployments are based on Standalone. In this study, we are focusing on the non-Standalone and also give a brief about Standalone support.

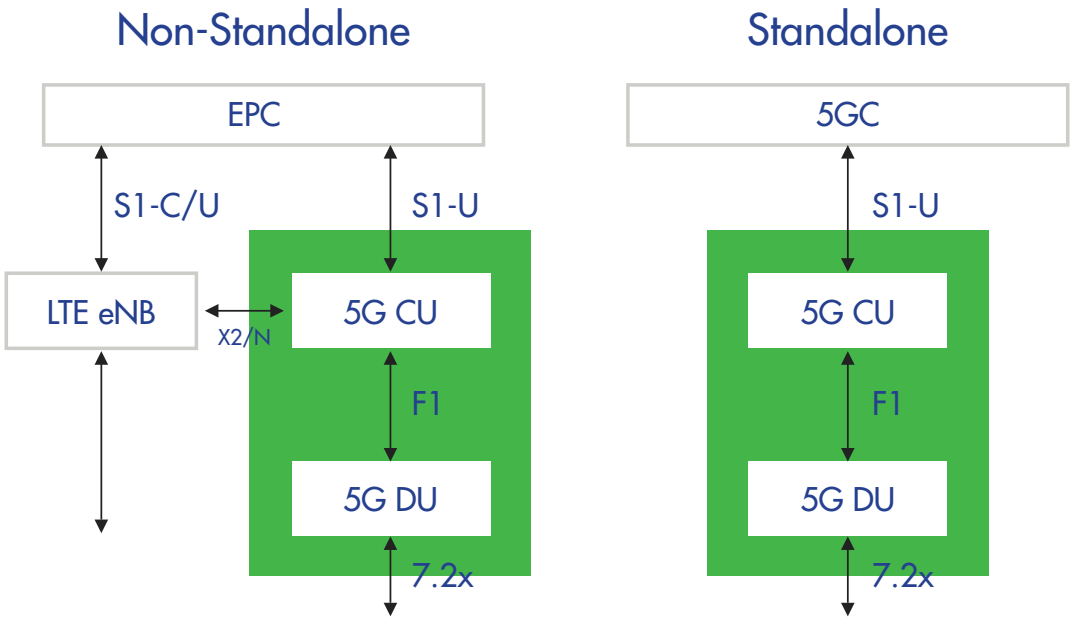


Figure 2: NSA and SA Deployment

For the current study in Radio access, below are the identified Open-Source components considered. These components are selected based on the availability and maturity of the software. The following two Open-Source components have very active community support. In the later part of the document, both of these components are evaluated in detail.

OpenAirInterface (OAI) software alliance, established in 2014 is a French non-profit organization, funded by corporate sponsors. The OSA is the home of OpenAirInterface, an open software that gathers a community of developers from around the world, who work together to build wireless cellular Radio Access Network (RAN) and Core Network (CN) technologies.

The Alliance is responsible for:

- the development roadmap.
- the quality control.
- the promotion of the OAI software packages, deployed by our academic and industrial community for varied use-cases.

The Alliance's mission is to facilitate OpenAirInterface adoption.

O-RAN Software Community (O-RAN SC) is a collaboration between the O-RAN Alliance and Linux Foundation with the mission to support the creation of software for the Radio Access Network (RAN). The RAN is the next challenge for the Open-Source community. The O-RAN SC plans to leverage other LF network projects, while addressing the challenges in performance, scale, and 3GPP alignment.

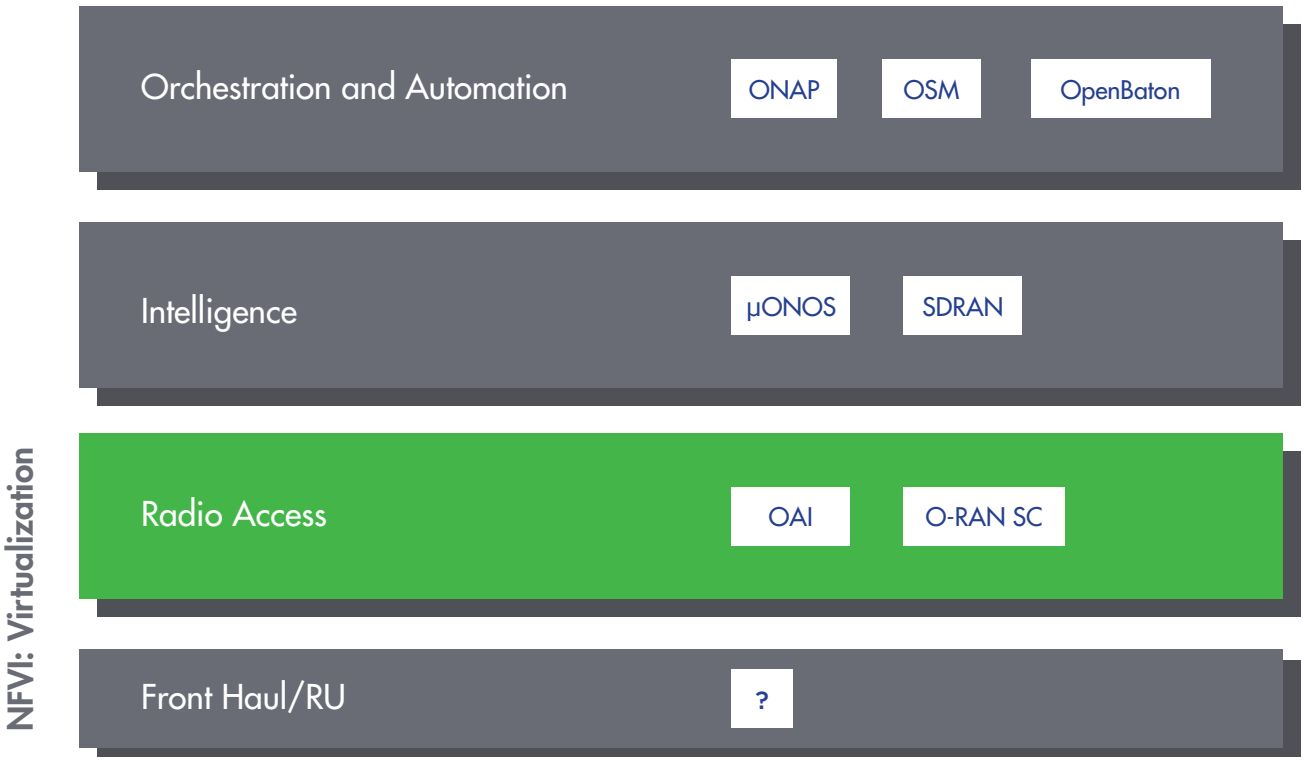


Figure 2: Open-Source Options for Radio Access

1.2 Evaluation Criteria

The activity also involves evaluating the Open-Source component's maturity. Below is the standard format used for the evaluation.

Criteria	Details
ORAN Compatibility	Adherence to ORAN specifications
License Type	The sort of license applicable
Full Software Availability	Access to full software in CU and DU
Software Quality	Software checks as per standard industry practices
Interoperability	Interoperability with third party components (Commercial and Test)
Roadmap to 3GPP Features	Path to future 3GPP releases
Commercial deployment considerations	Experience of Open-Source ecosystem (current and past) in commercial usage
History of ecosystem presence	Components in previous RAT technologies
Automation and CI	Standard automation and CI frameworks to maintain the software quality

1.3 Document Organization

The rest of the document is organized as follows:

- Section 2 describes the Requirement Specifications.
- Section 3 describes the Architecture.
- Section 4 describes the Evaluation.
- Section 5 describes the Conclusion.
- Section 6 describes the References.

2 - Requirement Specification

CU and DU software requirements are primarily driven by the deployment model chosen. For details on the deployment model considered, see the below sub-section. Considering the deployment scenario, CU and DU functional requirements are identified.

DU is the Distributed Unit that sits close to the RU and runs the RLC, MAC, and parts of the PHY layer. This logical node includes a subset of the gNodeB functionality, depending on the functional split option, and its operation is controlled by the CU.

Functional requirements are also further driven based on the feature set that is required. This is further detailed out in the Feature Requirement section.

Requirements for DU include compliance to below 3GPP specification requirements:

- 3GPP specification 38.321 - MAC.
- 3GPP specification 38.322 – RLC.
- 3GPP specification 38.212 - PHY.
- 3GPP specification 38.213 - PHY.
- 3GPP specification 38.214 – PHY.
- 3GPP specification 38.473 - F1AP.

CU is the Centralized Unit that runs the RRC and PDCP layers. The gNodeB consists of a CU and one DU connected to the CU via F1-C and F1-U interfaces for CP and UP respectively. A CU with multiple DUs will support multiple gNBs. The split architecture enables a 5G network to utilize different distribution of protocol stacks between CU and DUs depending on mid-haul availability and network design. It is a logical node that includes the gNB functions like transfer of user data, mobility control, RAN sharing (MORAN), positioning, session management etc., with the exception of functions that are allocated exclusively to the DU. The CU controls the operation of several DUs over the mid-haul interface.

Requirements for CU include compliance to below 3GPP specification requirements:

- 3GPP specification 29.281 - GTP.
- 3GPP specification 38.323 - PDCP.
- 3GPP specification 38.324 – SDAP.
- 3GPP specification 38.473 – F1AP.
- 3GPP specification 38.331 – RRC.
- 3GPP specification 38.413 – NGAP.
- 3GPP specification 38.423 – XNAP.

2.2 Non-functional Requirement

2.2.1 Hardware Requirements

There is no hardware requirement specified as part of this activity. It is proposed that the software be profiled with a reference configuration and based on that, arrive at the required hardware configurations for deployments.

Reference hardware configuration for initial tests is already provided by the Open-Source vendor.

2.2.2 Performance/Dimensioning Requirement

Hardware requirements are primarily driven based on the below factors.

- Number of Cells.
- MIMO Configuration.
- Bandwidth of operation.
- Max Data rates achieved for the configuration.
- Number of Users (Active/Connected).
- Duplexing mode.
- Loading Factor.
- Acceleration offload (Using Accelerators for Security, FastPath, L1 processing, IPSec etc.).

2.2.3 Scalability Requirement

Software should be scalable for below:

- Number of UEs.
- Number of CELLs.
- Higher Bandwidths.
- Performance headroom for feature additions (Up to Release 18).

2.2.4 Management Related Requirements

- Logging.
- Extendibility.
- Flexibility.
- Design and coding guidelines and principles.

2.3 Deployment Considerations

2.3.1 Deployment Scenarios

Deployment in the Radio Access space is primarily aligned to the ORAN architecture. Though 3GPP specifies various split options for the deployment, most industry deployments have aligned to the standard PDCP-RLC split and 7.2x splits.

Option 2 (PDCP/RLC split): Option 2 may be a base for an X2-like design due to similarity on U-plane but some functionalities may be different e.g., C-plane since some new procedures may be needed. There are two possible variants available in this option.

Split Option 7-2x: is one of the famous LLS split options adopted by ORAN fronthaul specifications. This functional splitting between Distributed Unit (O-DU) and Radio Unit (O- RU) divides the function of Physical Layer (Layer 1) named as High Physical resides in DU and Low Physical resides in RU.

Diagram in Figure 4 depicts the same.

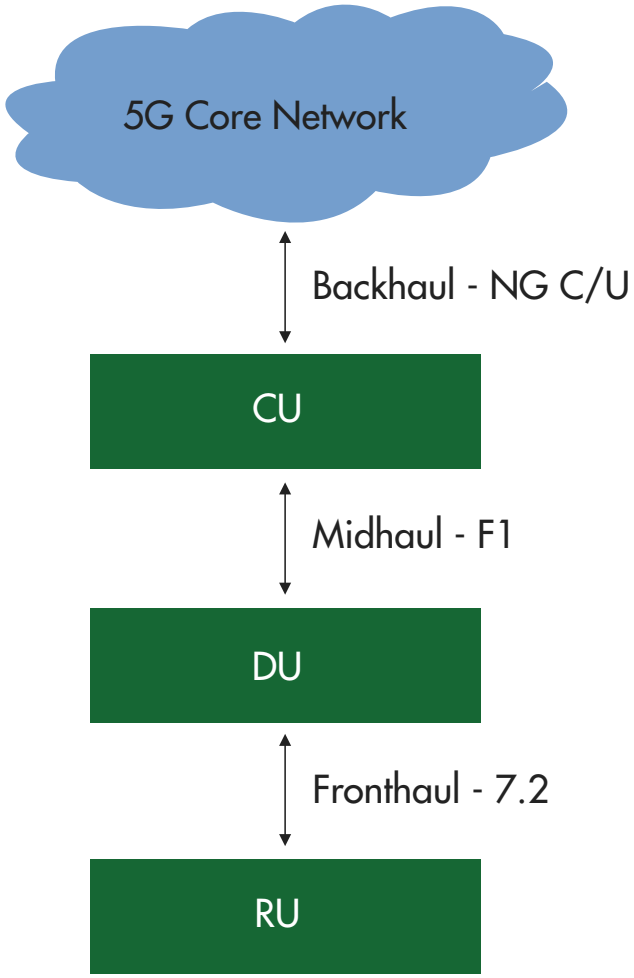


Figure 4: Split Options

3 - Architecture

3.1 System Architecture

3.1.1 End-to-end System Architecture

Below diagram presents a full view of the system showing the breakup of the components that we are focusing on, in the current study. The architecture is aligned to the O-RAN specification. The O-RAN is also shown below

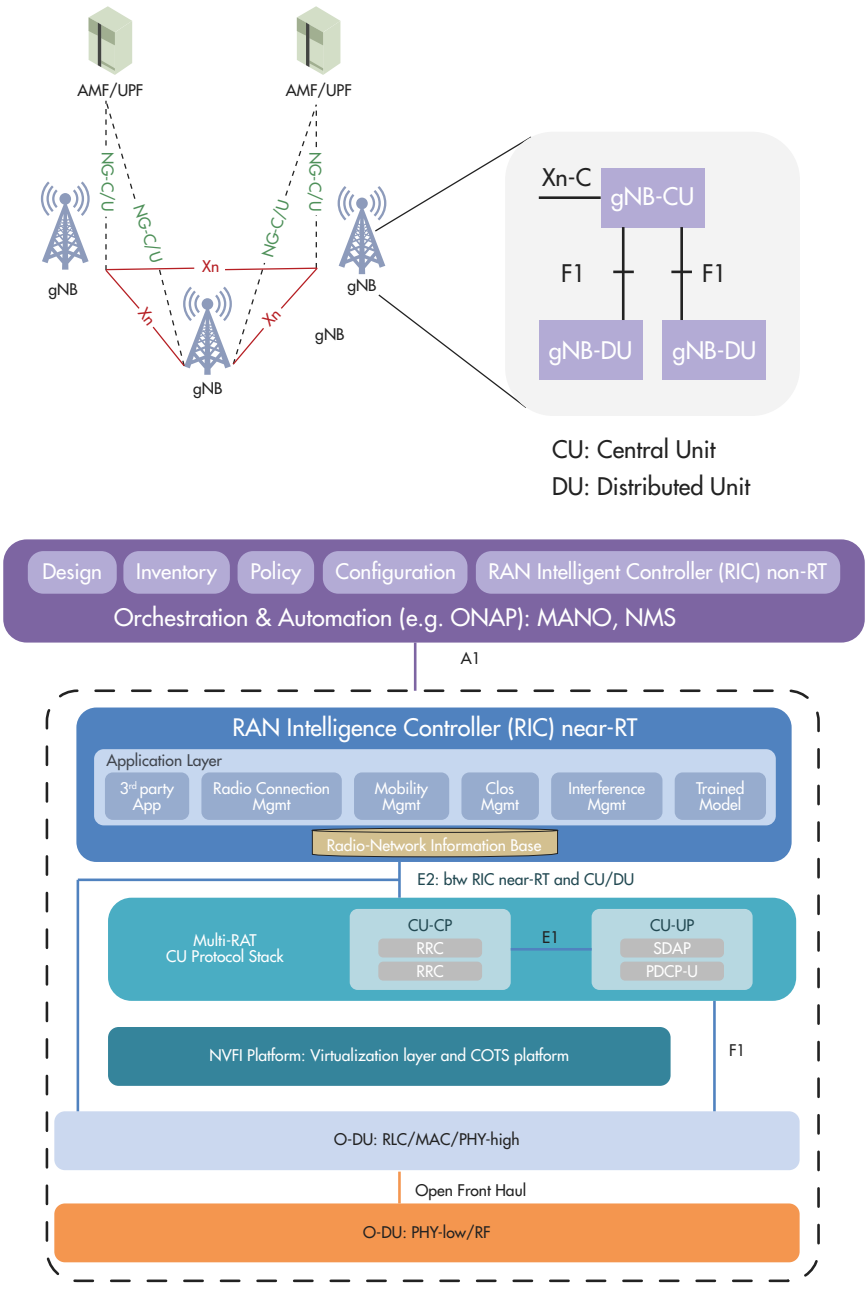


Figure 5: *End-End Architecture and ORAN Model*

3.1.2 Components Involved in System

For the current study, we are focusing on the CU and DU.

4 - Evaluation

4.1 Open-Source Evaluation for CU and DU

4.1.1 System Architecture Supported by Open-Source

a. **ORAN – SC:** The scheme below depicts the typical ORAN 5G architecture with NSA deployment.

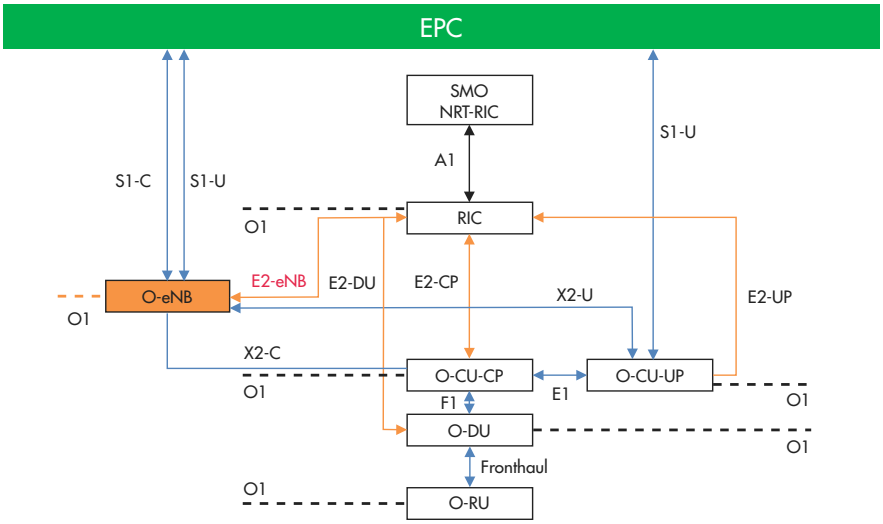


Figure 6: ORAN-SC System Architecture

b. **OAI:** The scheme below depicts the typical OAI NSA architecture with NSA and SA deployment.

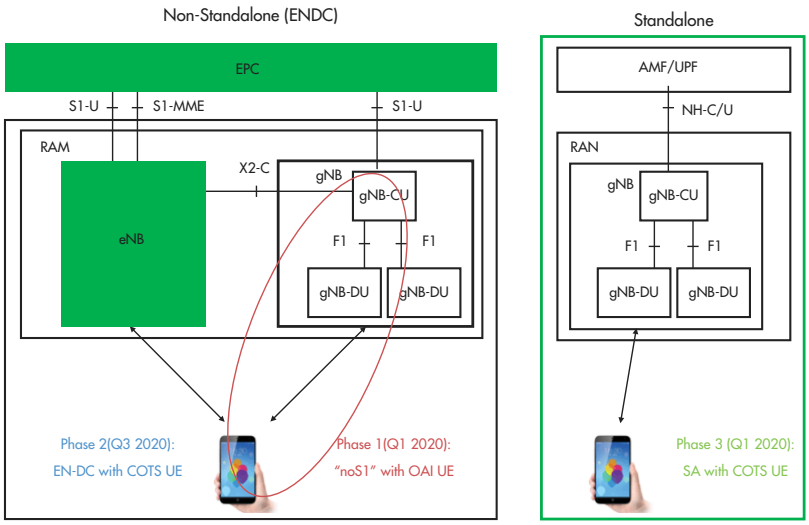


Figure 7: OAI System Architecture

4.1.2 Configuration Overview

a. ORAN – SC:

- Non-Standalone (NSA) configuration: Initial Control Plane is established between UE and RAN O-eNB, then User Plane is established between UE and O-gNB, Core network is 4G based
- Commercial UE
- ORAN software
- TDD
- FR1
- 100 MHz
- Antenna Scheme: SISO/MIMO/mMIMO/SU-MIMO
- Open-Radio Unit (O-RU), Open-Distributed Unit (O-DU), Open-Control Unit (O-CU), RAN Intelligent Control (RIC), Service Management and Orchestration (SMO) O-RU implemented as sample application running on the loopback mode

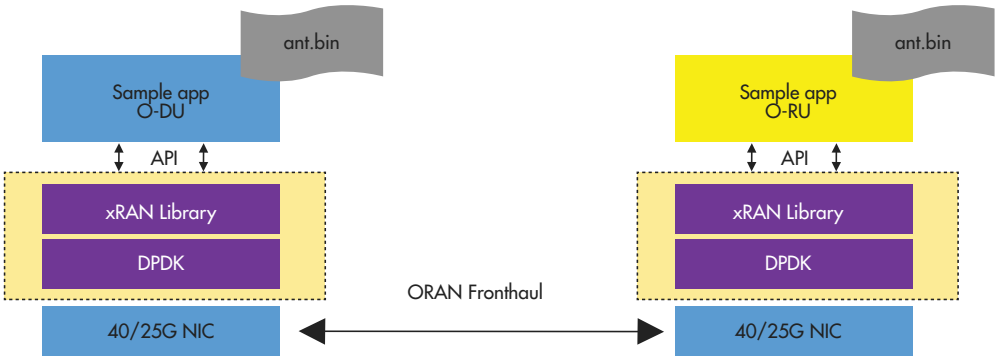


Figure 8: ORAN Deployment for Tests

b. OAI:

- Non-Standalone (NSA) configuration: Initial Control Plane is established between UE and RAN eNB, then User Plane is established between UE and gNB, Core network is 4G based supporting rel 15.
- Commercial UE: Oppo Reno 5G
- OAI Software Defined gNB and eNB
- eNB RF front end: USRP (ETTUS) B200 Mini or B210
- gNB RF front end: USRP (ETTUS) B200 Mini or B210 (N310 will be needed for MIMO and wider BW's)
- 5G TDD duplexing mode
- 5G FR1 Band n78 (3.5 GHz)
- BW: 80MHz (100MHz with newer USRP series)
- Antenna scheme: SISO

4.1.3 Feature Set Supported

a. O-RAN – SC:

O-DU:

Implementation of cell broadcast procedure and UE attach procedure (SA mode) for FD mode and FR1 (Numerology = 0, Bandwidth = 20 MHz) and basic scheduler APIs for single UE and single HARQ transmission.

- Initial UL RRC Message Transfer.
- UL/DL RRC Message Transfer.
- UE Context Setup Request/Response.
- F1AP Setup Request/Response.
- GNB DU Config Update.
- Basic FAPI messages implementation.
- UE attach procedure with basic scheduling on FDD, mu=0, BW=20 MHz.
- Single UE DL and UL data path and bench-marking.
- Support for 64QAM modulation scheme in DL and 16QAM in UL.
- Support for all short PRACH formats.
- Integrate O-DU High with O-DU Low.
- Integrate with Viavi sim/O-CU.
- Establish Netconf session for O1 interface for CM.
- Support Health Check use-case.

O-CU:

- RRC:
  - » Support Broadcast of system information.
  - » Support RRC connection control.
- NG:
  - » Support PDU Session Management Procedures.
  - » Support UE Context Management Procedures.
  - » Support Transport of NAS Messages Procedures.
  - » Support Interface Management Procedures.
- F1:
  - » o Support Interface Management procedures.
  - » o Support UE Context Management procedures.
  - » o Support RRC Message Transfer procedures.
  - » o Support System Information Procedures.

- E1:
  - » Support Interface Management procedures.
  - » Support Bearer Context Management procedures.
- SDAP
  - » Support transfer of user plane data.
  - » Support mapping between a QoS flow and a DRB for both DL and UL.
  - » Support marking QoS flow ID in both DL and UL packets.
  - » Support reflective QoS flow to DRB mapping for the UL SDAP data PDUs.
- PDCP
  - » Support transfer of data (user plane or control plane).
  - » Support maintenance of PDCP SNs
  - » Support header compression and decompression using the ROHC protocol.
  - » Support ciphering and deciphering.
  - » Support integrity protection and integrity verification.
  - » Support timer based SDU discard.
  - » Support reordering and in-order delivery.
  - » Support out-of-order delivery.

b. OAI:

The following features are valid for the gNB and the 5G-NR UE:

i. General Parameters

- » Static TDD.
- » FDD.
- » Normal CP.
- » 30 kHz subcarrier spacing.
- » Bandwidths up to 80MHz (217 Physical Resource Blocks).
- » Intermediate downlink and uplink frequencies to interface with IF equipment
- » Single antenna port (single beam).
- » Slot format: 14 OFDM symbols in UL or DL.
- » Highly efficient 3GPP compliant LDPC encoder and decoder (BG1 and BG2 supported).
- » Highly efficient 3GPP compliant polar encoder and decoder.
- » Encoder and decoder for short blocks.
- » Support for UL transform precoding (SC-FDMA).

ii. gNB PHY Layer

Below are the feature set supported by gNB PHY:

- 30KHz SCS for FR1 and 120 KHz SCS for FR2.
- Generation of NR-PSS/NR-SSS.
- NR-PBCH supports multiple SSBs and flexible periodicity.
- Generation of NR-PDCCH for SIB1 (including generation of DCI, polar encoding, scrambling, modulation, RB mapping, etc).
- Common search space configured by MIB.
- User-specific search space configured by RRC.
- DCI formats: 00, 10.
- Generation of NR-PDSCH (including Segmentation, LDPC encoding, rate matching, scrambling, modulation, RB mapping, etc).
- Single symbol DMRS, DMRS-TypeA-Position Pos2, DMRS configuration type1.
- PDSCH mapping type A.
- NR-CSI Generation of sequence at PHY.
- NR-PUSCH (including Segmentation, LDPC encoding, rate matching, scrambling, modulation, RB mapping, etc).
- NR-PUCCH.
- Format 0 (2 bits, mainly for ACK/NACK).
- Format 2 (up to 64 bits, mainly for CSI feedback).
- NR-PRACH.
- Formats 0,1,2,3, A1-A3, B1-B3.
- Highly efficient 3GPP compliant LDPC encoder and decoder (BG1 and BG2 are supported).
- Highly efficient 3GPP compliant polar encoder and decoder.
- Encoder and decoder for short block.

iii. gNB X2AP

Below are the feature set supported by gNB X2AP:

- X2 setup with eNB.
- Handling of SgNB Addition Request/Addition Request Acknowledge/ Reconfiguration Complete.

iv. gNB RRC

Below are the feature set supported by gNB RRC:

- NR RRC (38.331) Rel 15 messages using new asn1c.
- Generation of CellGroupConfig (for eNB) and MIB.
- Application to read configuration file and program gNB RRC.
- RRC can configure PDCP, RLC, MAC.

v. gNB MAC

Below are the feature set supported by gNB MAC:

- MAC -> PHY configuration using NR FAPI P5 interface.
- MAC <-> PHY data interface using FAPI P7 interface for BCH PDU, DCI PDU, PDSCH PDU.
- Contention-free random-access procedure.
- Support for multiple PRACH occasions.
- MAC uplink/downlink scheduler.
- MAC header generation (incl. timing advance).
- ACK/NACK handling and HARQ procedures for downlink/uplink.
- CSI measurement reporting.
- UL power control (PUCCH and PUSCH).
- Scheduler procedures for SIB1.
- Scheduler procedures for RA.
- MAC downlink scheduler (fixed allocations)
- As of May 2020, only DL was validated with COTS phone; UL in progress, validated with OAI UE in noS1 mode

4.1.4 Hardware requirements by Open-Source

a. ORAN – SC:

The below minimum hardware requirements are for setting O-DU and O-CU NSA system:

- O-DU High
  - CPU: 4 cores, RAM: 8G, Disk: 500G, NICs: 1.
- O-DU Low
  - Xeon® series Processor with Intel Architecture are supported and the platform should be either Intel® Xeon® SkyLake or CascadeLake with at least 2.0 GHz core frequency.
  - FPGA/ASIC card for FEC acceleration that is compliant with the BBDev framework and interface.
  - Running with FH requires PTP for Linux version 2.0 (or later) to be installed to provide IEEE 1588 synchronization.
- Interoperability Setup
  - Band n78 RRU for NR and Band B40 RRU for LTE.
  - Server to implement 4G BBU, 5G O-CU and EPC.
  - Antennas.
  - Commercial UE.

b. OAI:

The minimal hardware requirements are for setting up the OAI NSA system (Integrated CU/DU deployment). If CU-DU split deployment is desired, ORAN-SC similar hardware can be used.

- gNB RRC

The processing requirements for 5G-NR are much higher than for 4G, so a high-end PC or server is needed, like:

» Intel Core i7 6900K (8 cores), 16GB DDR, 480GB SSD Allows SW LDPC on 3 cores (1 segment per slot, 3 slots decoded in parallel, upto 30Mb/s) or LDPC on FPGA (up to 300Mb/s on 80MHz SISO).

» Intel Core i9 7980EX (18 cores) shall support future configurations incl. MIMO allowing parallel LDPC SW decoder on 9 or 12 cores, up to 3 (tested) or 5 (ongoing) segments per slot or 140Mb/s or LDPC on FPGA (same as above).

» Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz with 18 cores and 2x10Gbit Ethernet.

- SDR (Software Defined Radio) / USRP (Universal Software Radio Peripheral)

The processing requirements for 5G-NR are much higher than for 4G, so a high-end PC or server is needed, like:

» USRP N310

- This is the latest version of the USRP designed specifically for 5G-NR. It will support up to 100MHz bandwidth.

» USRP X310

- This older platform will also work with 5G-NR, but only supports bandwidths up to 80MHz with 3/4 sampling.

» USRP B210

- This platform can be used for bandwidths up to 40MHz using 3/4 sampling.

» SYRTEM platform

The SYRTEM platform is based on:

- The Xilinx EVALUATION KIT ZYNQ-7000 ZC706 (ex. supplier Digikey ref: 122-1904-ND, 2427€), and
- The Analog Device Transceiver type ADRV9371 (ex. supplier: Digikey ADRV9371-W/PCBZ-ND ('WIDE TUNING RANGE 300MHZ-6HGZ' version) 1169€).

- LDPC Offload

  As a high-performance alternative to the software LDPC decoder included in this release, the decoder can also be offloaded to another FPGA board (EVALUATION KIT ZYNQ - 7000 ZC706).

  The FPGA binary image is provided by Creonic, while the drivers are provided by Syrtem

- Interoperability Setup

  Below figure depicts the NSA interoperability setup:

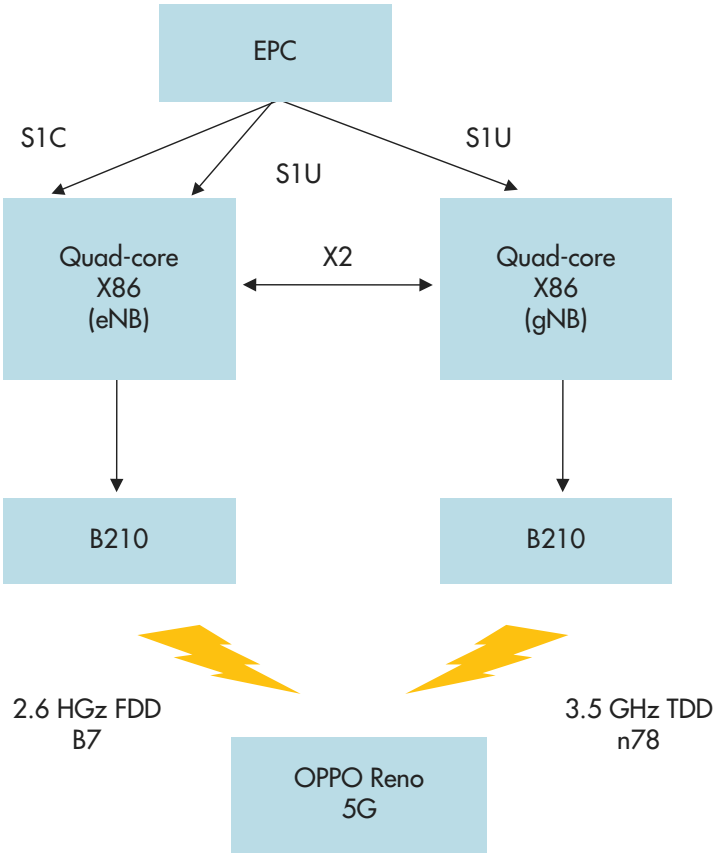


Figure 9: NSA interoperability deployment for OpenAirInterface

This interoperability NSA uses:

» Available EPC on network.
» Two PCs -
  - One eNB + USRP B210 (5/10 MHz LTE, Band 7)
  - One gNB + USRP B210 (40MHz NR, 30KHz SCS, TDD, Band n78)
» X2 for EN-DC over Ethernet between two PCs.
» Oppo Reno 5G (Qualcomm chipset)
» Qualcomm QCAT analysis software.

4.2 Current Limitations

a. ORAN – SC:

- Currently it supports 3GPP release 15, and release 16 is yet to be supported
- The complete ORAN specification is yet to be evolved. Phase I features are targeted by July'21 whereas Phase II features by November'21
- Build commercially viable solutions that meet high performing KPI requirements that support real-time system needs.
- Leverage adjacent software communities and open approaches to utilize existing solutions to speedup time to market.

b. OAI:

- Currently throughput is limited. UE does not decode MCS > 24.
- Currently scheduling is limited to a few slots. Not all the slots are being scheduled.
- Currently only single-user is supported. Scheduler to be upgraded in order to support multiple users.
- Currently there is no interoperability with FR2. Phone fails to see the 5G cell during Inter-RAT measurements.
- Adaptation to ORAN specifications ongoing

4.3 Gap Analysis

Area	Work
Algorithm	Scheduler algorithms (PFS, Round-robin etc)
Physical Layer	Physical layer algorithm optimizations (Beamforming, Channel Estimation etc)
Standardization	Impacts of system performance with ORAN compliance
Software Quality	Software checks as per standard industry practices
	Identify potential changes to the 3GPP specifications for ORAN deployments

Area	Work
Features	Numerologies and Bandwidths
	Physical Layer processing offload using acceleration cards
	Userplane accelerations
	ORAN compliance development

Area	Work
	Power control algorithms
	Advanced MIMO configurations
	Multiple Sector implementations
	Interference related features (ICIC)
Performance	Software profiling
	Multiple UEs
	Hardware dimensioning for capacity requirements
Interoperability	Test with Industry standard commercial devices
	Test with standard Core Networks
	Test with third-party LTE NSA systems
	Test with third-party Radios
	Integration with ORAN compliant components

4.4 Roadmap

a. ORAN – SC:

Below lists out the road map for NSA:

- EN-DC support.
- Basic beam-forming supported for FR2.
  - » Initial access.
  - » CSI feedback.
  - » TCI states.
- Advanced scheduler for gNB
  - » This is required for multiuser scheduling.
- Basic FR2 Interoperability.
- MIMO support.
- Standalone Implementation
  - » Below features expected on D release – Jul 2021.
  - » ORAN Central Unit.
  - » Radisys Commercial CU being used as a test fixture for E2E testing.

» In the absence of O-CU, Radisys commercial CU image to be used for E2E testing.

ORAN Distributed Unit High

» Achieve UL and DL data flow using FDD mode on 20 MHz Bandwidth.
» Numerology = 0.
» Support for static TDD mode with pattern "DDDDDDDSUU" on 100 MHz Bandwidth, Numerology = 1.
  - Evolve scheduler to support UL and DL scheduling of signalling and data messages on single spectrum in TDD mode.
  - Expand scheduler to support Frame structure according to numerology = 1.
  - Updates to cell broadcast for TDD and numerology = 1.
» Development activity for Closed Loop Automation use-case
  - Support for cell stop and restart within O-DU High layers.
  - Support for cell stop and restart towards O-DU Low.
  - F1AP Enhancements towards O-CU indicating cell stop and restart.
» Integration
  - Integration with O-DU Low in Radio mode.
  - Integration with CU.

- Network Slicing Package

  Phase I: O-CU and O-DU Slicing related IMs, Dynamic deployment of ORAN NFs through O2 interface, End-to-end definition of RAN Slice SLA assurance use case, including Non-RT RIC and Near-RT RIC, Slice SLA A1 policies.

  Phase II: It will be delivered in November 2021, builds up on the first phase features with the following: Further definitions of O-RAN slice subnet management use cases, Extended set of SLA parameters and A1 policies, Extension of O-CU and O- DU Slicing related IMs, E2SM support for Slice SLA Assurance use case, Dynamic and orchestrated mapping of intra-DC slice links to inter-DC transport links.

b. OAI:

- Standalone:

  The progress done so far on SA gNB and also the road-map are as follows:
  » o PHY.
    - Initial bandwidth part & CORESET 0 (To be implemented).

» MAC
  - Configuration of CORESET 0 through MIB (Completed)
  - Scheduling of SIB 1 using CORESET 0 (Completed)
  - Contention resolution random access (To be implemented)
» RLC
  - Support for SRB (To be implemented)
» F1 interface (To be implemented)
» RRC
  - Messages (Completed)
  - Configuration of RLC/MAC/PHY (Completed)
  - UL/DL information transfer (To be implemented)
  - Interface with NGAP (To be implemented)
  - Interface with PDCP (To be implemented)
» NGAP
  - Messages (Partially completed)
  - Interface with AMF (N2/SCTP) (Partially completed)
  - Interface with RRC (To be implemented)
» SDAP
  - SDAP Header insertion (To be implemented)
  - Interface with UPF (N3/GTP-U) (To be implemented)
» Architecture Improvements
  - GTP-U (To be implemented)
  - RRC (To be implemented)

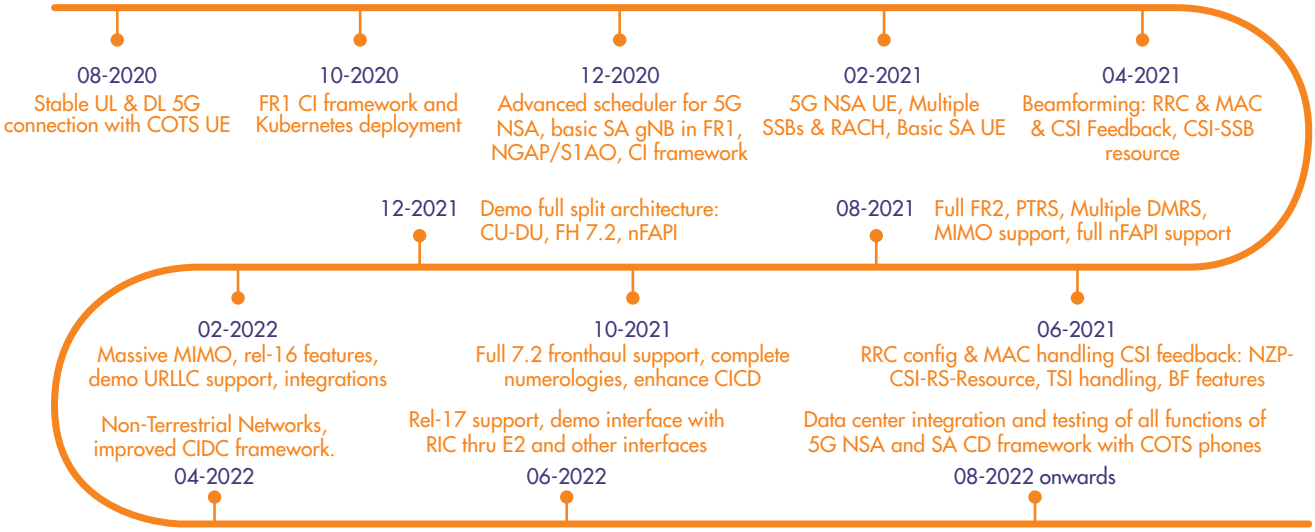


Figure 10: OAI Roadmap – Part 1

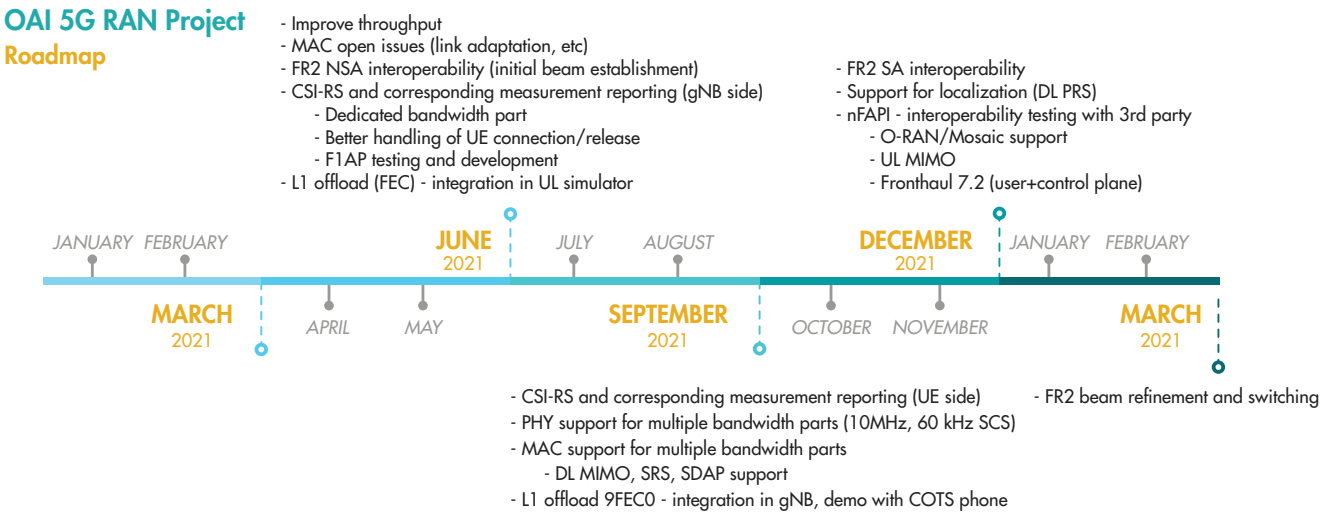


Figure 11: OAI Roadmap – Part 2

4.5 Comparison of different Open-Source

Based on the evaluations done with respect to above factors, below is the recommendations for CU and DU software:

Criteria	Details	OAI	ORAN-SC
ORAN Compatibility	Adherence to ORAN specifications	Yes	Yes
License Type	Sort of License applicable	OAI Public License V1.1 (modified version of Apache V2.0)	ORAN SC license (modified version of Apache V2.0)
Full Software Availability	Access to full software in CU and DU	Yes	Partial
Software Quality	Software checks as per standard industry practices	Tools Available	Tools Available
Interoperability	Interoperability with third party components (Commercial and Test) Path to future 3GPP releases	Yes	Partial
Roadmap to 3GPP Features	Path to future 3GPP releases	Yes	Partial
Commercial deployment considerations	Experience of Open-Source ecosystem (current and past) in commercial usage	Lab Trials Ongoing	Yet to start

Criteria	Details	OAI	ORAN-SC
History of ecosystem presence	Components in previous RAT technologies	Yes	5G only
Automation and CI	Standard automation and CI frameworks to maintain software quality	Yes	Ongoing

5 - Conclusion

5.1 Recommended Open-Source

Given the evaluation criteria, a detailed study of the software was done from both the ecosystems both the ecosystem and below is the conclusion.

Component	Open-Source Option
CU	OpenAirInterface
DU	OpenAirInterface

5.2 Minimum Viable Product

- Replicate the OpenAirInterface setup.
- Test for performance bench-marking as declared by OpenAirInterface.
- Test with standard commercial UEs and CoreNetwork.
- Test for a greater number of UEs and different configurations.
- Enhance the OpenAirInterface software for ORAN compatibility (CU-DU split, CU Split, O1 and other interfaces).
- Test with standard ORAN 3rd party components.
- Arrive at the hardware dimensioning required for scalable deployments (Inputs could be arrived from any operator for deployment scenario).
- Benchmark on the new hardware as per the reference operator requirement.
- Identify critical feature set in discussion with operators for deployment.
- Implementing those missing feature sets in the OpenAirInterface.

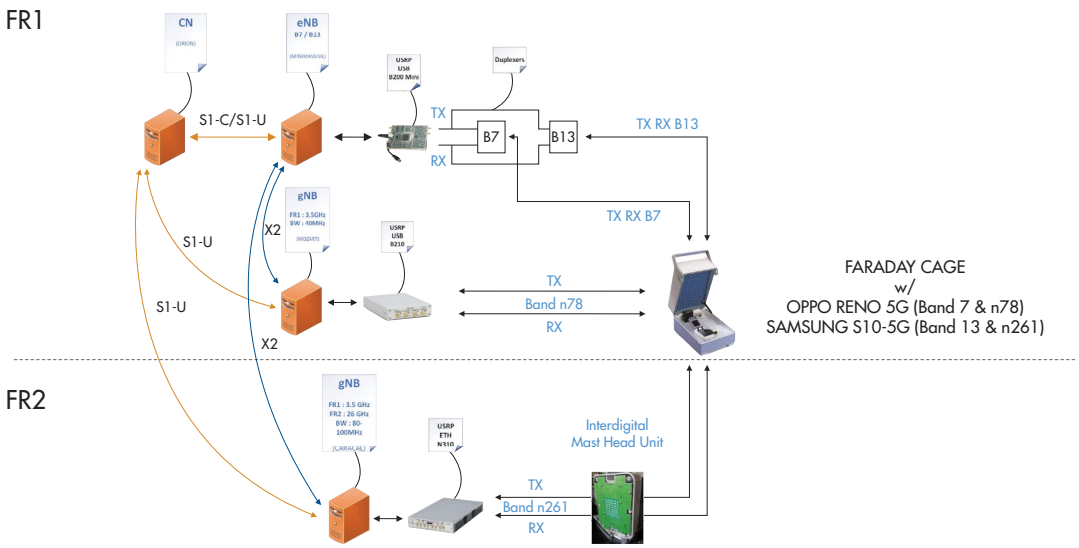


Figure 12: OAI PoC Setup

5.3 Future Work

From the study there has been a lot of work items identified for taking the product into field trials and deployments. Further there is a major gap in taking the software to full 5G compliance as per the 3GPP. Major areas of work are as below:

- Full Network slices support: This is for supporting mMTC and uRLCC majorly.
- Virtualization: To leverage the softwarization of 5G Open RAN architecture.
- Porting the platform to different platforms for bench-marking (NXP, Marvell): To enable the ecosystem to have a level ground platform to decide on the best cost- effective platforms for deployments.
- Harmonization of O-RAN and 3GPP specification in the full virtualized deployment: There are gaps in both specifications to make the systems fully ready for virtualized deployments. Identify them and propagate them into 3GPP specifications.

6 - References

- ORAN SC (https://docs.o-ran-sc.org).
- OAI (https://www.openairinterface.org).

Transport Sub-Team

1 - Introduction

This addendum is the report of the Transport Sub-Team within the Task-Force and supports the "Feasibility of Open-Source for 5G- Final Overall Report" issued by the TSDSI Task-Force on Open-Source for 5G. Please read the main section to get the context, background and overall recommendations of the Task-Force.

Multiple organizations are looking to embrace Open-Source and disaggregated solutions in order to increase the flexibility and affordability of their network infrastructure. Open-Source softwares like Linux and Android have played a pivotal role in the democratization of the Server OS and Mobile handset OS. In the last decade, Disaggregated White Box solution has transformed the data centre market and is a dominant player.

The key benefits of Open-Source and disaggregated solution include:

- Brings diversity by community driven development with contribution from across the world including individuals, universities, enterprises, competitors.
- Fosters innovation through collaboration, community-led roadmap.
- Drive openness and reduce dependency on proprietary solutions.
- Brings in flexibility with simpler upgrades, easy integration to 3rd party application and hardware.
- Helps in reducing the Total Cost of Ownership (TCO).

In recent years, there is strong interest within the Telecom industry to reap the benefits of open and disaggregated solutions. Telecom operators have been driving and contributing to multiple Open-Source software and open hardware projects. Few telecom operators have already deployed Open-Source solutions within their network.

From an Indian ecosystem perspective, with 5G deployment on the horizon, this is the crucial and the right time to evaluate and consider the Open-Source options that are available.

1.1 Scope

The motive of the Transport Sub-team is to give a head start to the India ecosystem for choosing an Open-Source ecosystem for 5G Transport by evaluating the existing projects and their readiness for deployment.

The packet transport network is used to forward 5G traffic over an IP/MPLS backbone to the 5G Packet Core. The Transport sub-team mainly focused its study on 3 components:

- Disaggregated Cell Site Gateway (DCSG)
- Edge Router (ER)
- Software Defined Networking (SDN) Controller

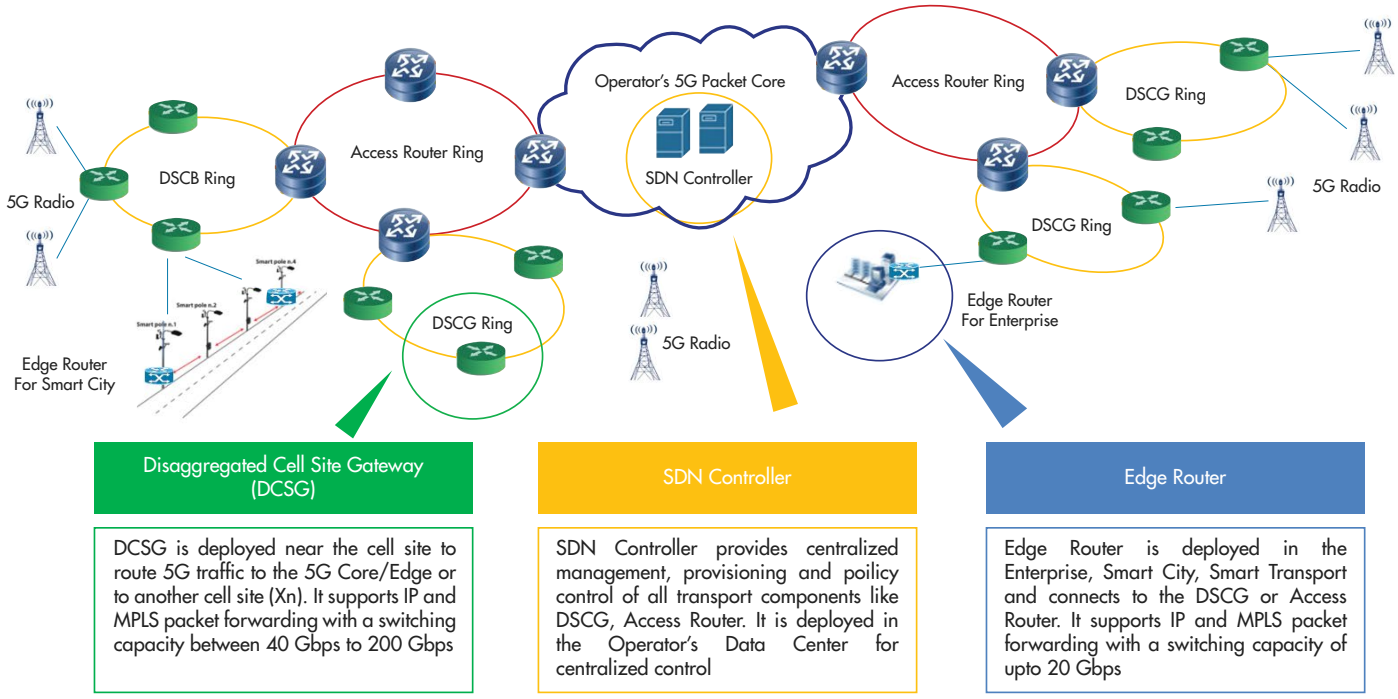


Figure 1: IP Transport Deployment

The DCSG is a critical component within the mobile operator's network that is deployed near the cell site and forwards 5G traffic over an IP/MPLS backbone toward the 5G Packet Core. DCSG also forwards inter-cell site 5G traffic (Xn). DCSG has a switching capacity between 40 Gbps to 200 Gbps. As the DCSG is deployed near the cell site, there would be 100s of thousands of DCSG deployed by the tier-1 operators.

ER is deployed within an Enterprise or for Smart City, Smart Transport. It is used to forward Enterprise traffic or 5G traffic over an IP/MPLS backbone. The ER is a miniature version of the DCSG with a switching capacity of less than 20 Gbps.

The SDN Controller would be used for the provisioning and management of all the IP Network nodes within the operator's network infrastructure. The SDN Controller is centrally deployed within the operator's data centre and manages all the DCSG, Access Routers etc.

1.2 Evaluation Criteria

We evaluated the Open-Source projects for Routing Stack and SDN Controller for its features, ease of management, regular release cycles, test/automation support and an active roadmap.

The following is the evaluation criteria for the Open-Source Routing Stack:

Criteria	Details
IPv4, IPv6 Support	Support IPv4 and IPv6 Protocols for Packet Forwarding
OSPF, ISIS, BGP, PIM	Support of Routing protocols for within an Autonomous domain (AS) or between AS.
MPLS, SR Support	Support MPLS protocols for Packet Forwarding
SRv6 Support	Support Segment Routing with IPv6 Packet Forwarding
L2 and L3 VPN	Support Layer 2 and Layer3 Virtual Private Networks using MPLS
6PE, 6VPE	Support Layer 2 and Layer3 Virtual Private Networks for IPv6
Traffic Engineering, SR-TE, PCE	Support for Traffic Engineering Capabilities to guarantee throughput, latency and network slicing
Layer 2 - Y.1731, LACP	Support of Layer-2 Capabilities
Timing Protocol – PTP, SyncE	Support of Timing Synchronization features
Netconf	Support of Netconf and yang models for centralized management and provisioning
CLI and Management	Support of CLI for Management
Test Automation and CI	Open-Source project support should have a strong test automation framework
Platform Support	Support for System platforms
Active Contributors	The project must have continuous commit, regular releases and active contribution from various participants

The following is the evaluation criteria for the Open-Source SDN Controller:

Criteria	Details
Types of Intents available and conflict resolution	Support for multiple management, intent and configuration options
Authentication/authorization support	Support of flexible AAA
NB/SB protocol options	Support of multiple Northbound and Southbound protocol options

Criteria	Details
Community activeness	The project must have continuous commit, regular releases and active contribution from various participants
HA architecture	Support a High Available System with Geo- redundancy for Disaster Recovery
Collaboration with other Open-Source communities	The project must have active collaboration with other Open-Source projects

1.3 Document Organization

The rest of the document is organized as follows:

- Section 2 covers the requirement and deployment topology
- Section 3 describes the System Architecture along with Software and Hardware design
- Section 4 evaluates the Open-Source Projects with Gap Analysis and Roadmap
- Section 5 recommends the Open-Source Software, Hardware and the PoC details

2 - Requirement Specification

This section covers the Functional Requirements, Hardware Requirement and deployment topology of DCSG, ER and the SDN Controller.

2.1 Functional Requirement

2.1.1 DCSG and ER Feature Requirement

Below are the requirements for DCSG and ER. ER is a miniature version of the DCSG that would not require advanced features like Segment Routing, Traffic Engineering, BGP and would need lower throughput.

			Requirement		
Sl. No	Requirements	Priority	DCSG	ER	Access
1	IGP protocol	Heading			
a	The device should support ISIS as IGP protocol	P1			

			Requirement		
Sl. No	Requirements	Priority	DCSG	ER	Access
b	The device should support OSPF as IGP protocol	P1			
2	The Device should support BGP/ MP BGP with following families	Heading			
a	IPv4/Ipv6	P1			
b	VPNv4	P1			
c	VPNv6	P1			
d	MVPN	P1			
e	EVPN	P2			
3	Device Should support the following protocols	Heading			
a	1588v2 (All Profile 827.1/.2 etc.)	P1		NA	
	Conversion of Timing Profile	P1		NA	
b	SR BE	P1		NA	
c	Ti-LFA	P1		NA	
d	SR-TE	P2		NA	
e	SRv6	P3		NA	
f	TACAS+/RADIUS	P1			
g	Port and VLAN based Traffic mirroring	P1			
h	IEEE 802.3ah with dying gasp support	P1			
i	TWAMP	P1		NA	
j	Y.1731	P1		NA	
k	LDP and rLFA	P1			
l	IP/MPLS FRR	P1		NA	
m	Netconf/Yang	P1			
n	BGP-LS	P2		NA	
o	PCEP	P2		NA	
4	Device should support PIM – SM and SSM	P2			
5	Device should support Management over Ipv6	P2			
6	The router shall support using any of the 4,095 802.1q VLAN values.	P1			

			Requirement		
Sl. No	Requirements	Priority	DCSG	ER	Access
7	The router shall support VLAN tag manipulation	P1			
8	Device Should support service Ingress Classification (DSCP and Dot1p, IP (IPv4 and IPv6), MAC)	P1			
9	Device Should support network Ingress classification based on MPLS EXP bits	P1			
10	Device Should support QOS marking for Self-generated traffic	P1			
11	Device shall support 50*3ms BFD for VPN, IPv4 and Ipv6 as well as micro-BFD	P1			
12	Device should support BGP-LU with Prefix Sid Index	P1		NA	
13	Device should support LDP and SR simultaneously	P1	NA	NA	
14	Device should be open for 3rd party Optics	P1			
15	Device should be Interoperable with 3rd Party devices over standard protocols	P1			
16	Device should support HVPN	P1		NA	
17	Device should support inline RR feature with ORF and RT filtering	P1	NA	NA	
18	Device should support PHTe (Pseudowire Headend Termination)	P1	NA	NA	
19	Device should support VRRP and MCLAG with control packets over Pseudowire	P1		NA	

2.1.2 SDN Controller Feature Requirement

Sr No.	High level Requirement	Notes
1	Devices can be deployed in IP/MPLS, Microwave and Optical networks. To support this, SDN controllers will follow a hierarchical architecture. There will be a hierarchical SDN Controller and domain SDN controllers for IP/MPLS, microwave or optical	Devices - DCSG The focus of this document is for the transport SDN controller. Hierarchical SDN Controller => Similar to a Network Service Orchestrator (NSO) dictating the service provisioning end to end via domain specific controllers
2	Hierarchical SDN Controller and domain SDN controllers will have a RESTCONF interface between them.	
3	Hierarchical SDN Controller will offer a RESTCONF based northbound interface to the OSS/BSS	
4	Domain SDN controllers will support vendor neutral YANG models from Openconfig, IETF, and ONF for NB interaction with hierarchical SDN controllers. For the transport part, the focus will be on IP network data models (L3VPN service NBI, L2VPN service NBI, L1/L2/L3 topology NBI, Traffic Engineering NBI). This enables end to end transport service programming and 5G network slicing automation	Requires SDN controller to add these YANG models and an internal app to convert the NB REST calls to SB APIs based on these YANG Models
5	Domain SDN controller will use management protocols like NetConf/REST/protobuf to interact with Devices, for configuration/provisioning	
6	Domain SDN controllers will support vendor neutral YANG models from OpenConfig, IETF and ONF for SB interaction with Devices. For the transport part, it will include modules to configure disparate protocols (e.g., BGP, LLDP, LACP), modules to configure local routing (e.g., to include static routes), modules to retrieve the hardware information (chassis, cards, ports, fans, CPUs, et al.), and modules to configure VRFs and create VPNs	Requires SDN controller to add these YANG models - Netconf protocol implementation should be able to fire edit-config to devices based on these

Sr No.	High level Requirement	Notes
7	Domain SDN controller will collect real time network data (using NetConf or gRPC/gNMI/SNMP) from the Devices and make it available to the hierarchical SDN controller via RESTCONF NBI Another alternative is to get telemetry from the existing NMS/EMS system. This can be done via XML/REST API calls to the NMS/EMS, based on what NB API the existing NMS/EMS provides	Support for gRPC/gNMI is still evolving on the Open-Source SDN controllers. SDN controllers will need apps which trigger subscribing to telemetry information For interworking with existing NMS/EMS, there might be considerable work to implement SB drivers based on XML/REST
8	Domain SDN controller will use NetConf to collect events and alarms from the devices and will use NBI to communicate the same to the hierarchical SDN controller Domain SDN controllers can also use the existing NMS/EMS to retrieve these events/alarms	SDN controllers will need a fault management app, which can get alarms from Devices and offer REST based NB API to retrieve these alarms. There are YANG models for Alarms as well (https://www.ietf.org/archive/id/draft- vallin-netmod-alarm-module-02.txt) though it's a draft version - so NetConf should be possible for this feature For retrieving alarms/events from existing NMS/EMS, NB interface of the NMS/EMS needs to be exposed
9	Domain SDN controllers will support provisioning of devices (in terms of VPNs, multicast TV, B2B, optical channels, L2 connections) using NetConf/REST/ protobuf SBI. These provisioning mechanisms will be required for SLA fulfilment and service provisioning at Devices domain	
10	IP/MPLS segment will also support BGP-LS to enable the domain SDN controller to retrieve the layer 3 and SR topology and will have PCEP to support TE and for reporting LSP statuses, instantiating PCE-initiated paths and modifying PCE-delegated LSPs	

Sr No.	High level Requirement	Notes
11	Domain SDN controllers will support Path computation element which will be triggered in two cases: 1) A new end to end service is created and hierarchical SDN Controller dictates the domain controller to create a part of it on its associated devices 2) Due to traffic optimizations triggered by the operator or traffic engineering application running in the domain SDN controller or the Hierarchical controller. This is generally based on the TE topology and real time network data. As per service provider requirement, the path computation element might run at the hierarchical SDN controller. The hierarchical SDN controller will have a PCE App taking decisions based on consolidated topology/telemetry from the domain SDN controllers	Traffic engineering apps are generally written on the top of Open-Source SDN controllers
12	Hierarchical SDN controller will offer NBI interface to OSS/BSS to retrieve the telemetry data and events/alarms, which it has collected from the domain SDN controllers. For a single pane of glass management, there will be a centralized fault management system which may or may not be co-located with the hierarchical SDN controller. This FMS will retrieve the FCAPS data from the domain SDN controller's NB interface (in terms of Alarms/ events or stats) or will get this data from the existing EMS per domain.	
13	The centralized fault management system will require an interface with one EMS/NMS per domain if all devices of that domain belong to the same vendor	

Sr No.	High level Requirement	Notes
14	The centralized fault management system will require an interface with one EMS/NMS per vendor per domain if devices of that domain belong to more than one vendor	Each vendor will have their own EMS systems. There might be a scenario where multiple EMS are connected to a single NMS. The centralized fault management system will have an interface to the NMS in this scenario. Scalability of EMS might trigger a transition from single EMS to multiple EMS-single NMS model
15	If devices of all domains (applies only to multi-vendor IP scenarios) offer a NB interface which the hierarchical SDN controller can leverage for provisioning, the service provider will not require domain controllers. The hierarchical SDN controller will be used for directly configuring the devices	Requires forward compatibility of the devices with the hierarchical SDN controller, in terms of protocol support
16	Hierarchical SDN controller and domain SDN controllers will operate in a cluster environment. Each domain SDN controller is basically multiple instances of an Open-Source SDNC, operating in a highly available mode	Most of the SDN controllers have an inherent support of clustering where multiple instances work - each managed device is connected to one of these instances and these instances sync data among themselves using east-west communication. If SDNC instance connected to a device goes down, another SDNC instance of the cluster triggers a SB connection with the device
17	Hierarchical SDN controller will support RBAC on NBI, for operators	Support for AAA and RBAC in Open-Source SDN controller is quite limited as of now
18	Domain SDN controller will support HTTPS enabled REST interface, for REST communication from the Hierarchical SDN controller	REST/RESTful
19	TLS will be enabled for the EAST-WEST communications between instances of the SDN controller running as a HA cluster	

2.2 Non-functional Requirement

This section covers the hardware requirement for DCSG. The ER would be a scaled down version of DCSG with a switching capacity of 20 Gbps and without the need for Clock Synchronization, Segment Routing and Traffic Engineering. SDN Controller should be able to run on a server or on a Virtual Machine (VM).

2.2.1 Hardware Requirements

Sr No.	Type	Heading	DCSG
1	**Heading**	**Physical Compliance**	
1.1	Scale	Size (RU)	1 U
1.2	Scale	Depth (mm)	300 mm
1.3	Scale	Dimensions of proposed Device	Yes
1.4	Scale	Full duplex throughput Required	60 Gbps / 32 Gbps
1.5	Scale	Full duplex Capacity per slot	Yes
1.6	Scale	Buffer Per Interface	Yes
1.7	Scale	Device Buffer	Yes
1.8	Feature	Device should have internal GNSS port(receiver) integrated and will connect to external antenna (should support GPS, Glonass, Galelio)	Yes
1.9	Feature	Control plane redundancy	No
1.10	Feature	Power Redundancy	Required
1.11	Scale	Please specify the Type of power Supported	DC
1.12	Feature	Proposed Device should be temperature hardened	Yes
1.13	Scale	Supported Working Temperature Range	-10 to +65 C
1.14	Scale	State MTBF values for the offered products	Yes
1.15	Scale	State MTTR values for the offered products	Yes
1.16	Feature	The router shall have front to back airflow	Yes
1.17	Feature	The router must be serviceable with having front access only	Yes
1.18	Feature	Specify if Power, Control Plan, Fan are hot swappable	Yes

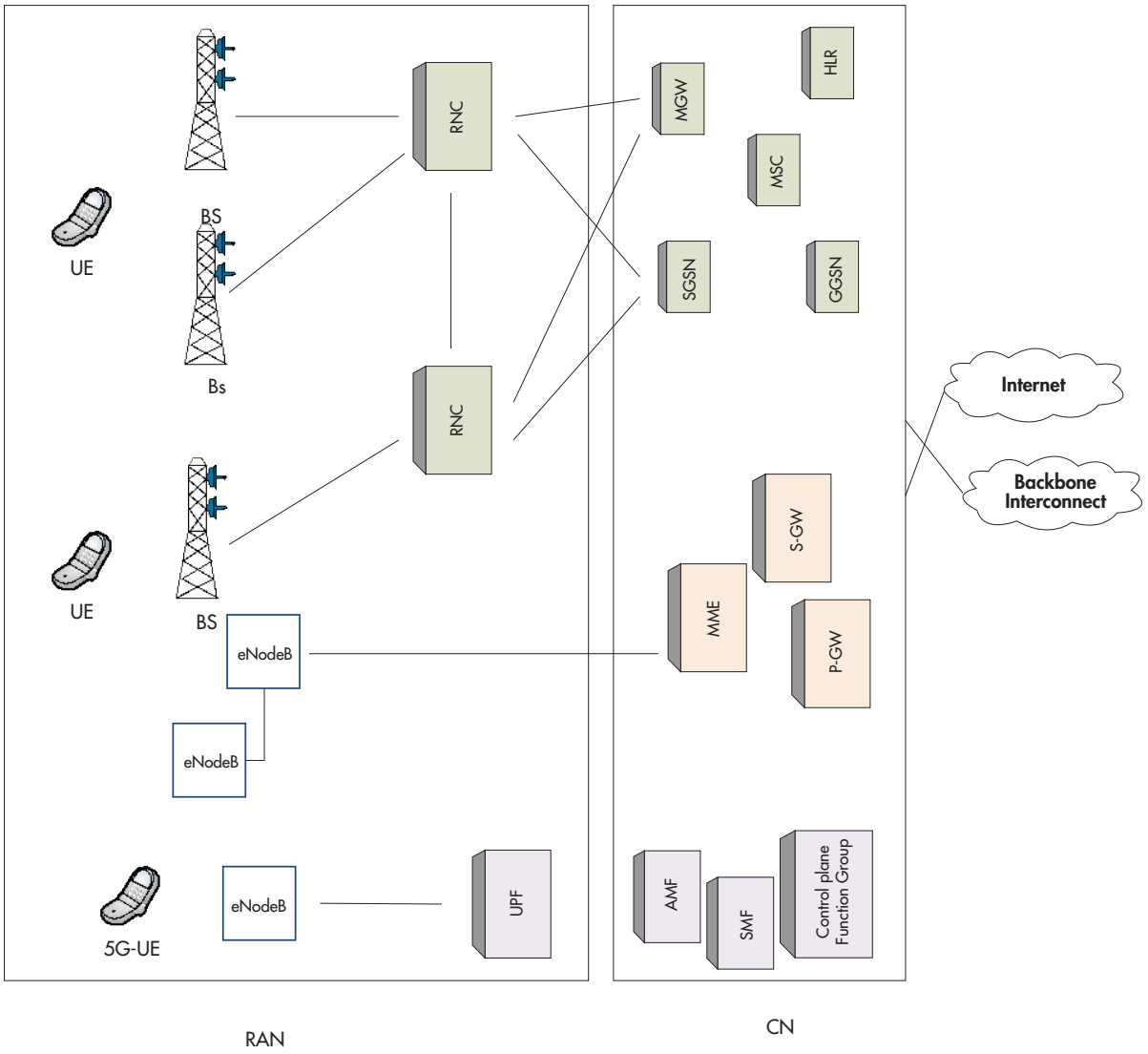
2	Heading	Supported Interface	
2.1	Feature	Support for Console Port	Yes
2.2	Feature	Support for Management Port	Yes
2.3	Scale	Minimum number of 1GE ports (optical) full rate Required	8
2.4	Scale	Minimum number of 10GE ports full rate required	4
2.5	Scale	Minimum number of 100GE ports full rate required	0
2.6	Feature	The router shall support at least one dedicated TOD/1PPS RJ45 interface for time/phase synchronization input or output	Yes
3	**Heading**	**Power Consumption**	
3.1	Scale	Typical power consumption (W)	Yes
3.2	Scale	Maximum Power consumption (W)	Yes
4	**Heading**	**HW compute CPU**	
4.1	Feature	ASIC Merchant silicon name	Yes
4.2	Feature	Manufacturer and model	Yes
4.3	Scale	Clock frequency (GHz)	Yes
4.4	Scale	Number of cores	Yes
4.5	Scale	RAM capacity (GB)	Yes
4.6	Scale	Storage (GB)	Yes
4.7	Feature	Coating standard on hardware	Yes
4.8	Feature	Clock Chipset Model (for 1558v2/SyncE)	Yes

2.3 Deployment Considerations

2.3.1 DCSG and ER Deployment Scenarios

The DCSG based router can be deployed at several places in the RAN and in the CN (Core Network) as Edge aggregation network. On the RAN side, the role and features of DCSG is expected to vary, depending on the wireless technology in use. As an example, if the deployment is for a 3G or 4G network, the DCSG is expected to play the role of a transport router and will be deployed close to the cell towers. The services that is primarily expected to provide is transport service for IP and timing services. If it is being deployed for a 5G network, the primary service it is expected to provide is, transport service for IP, Radio over packet and timing services. The size and form factor of the DCSG could again vary depending upon the deployment design. In a few cases DCSG would provide features enabling it for Microwave backhaul, and in some cases also act as provider edge device for enterprise customers.

The second case is in the CN, where DCSG is expected to be deployed at the aggregation points, the expected place for which would be at the core network edge aggregation point. While there could be other aggregation nodes requirements at the RAN level, the usage of them is subject to the design criteria. An example of using DCSG in the RAN would be for O-RAN architecture split where it could play the role of DU-AR (Distributed Unit-Aggregation Router) and CU-AR (Central Unit – Aggregation Router).



DSCG in Use

Figure 2: Generic Deployment Points for DCSG in Various Mobile RAN/CN Technology

The DCSG connectivity would mainly be in one of the 4 combinations – Single Homed Ring, Dual homed Ring, Hub & Spoke/Spur and Daisy Chain. The ER would be deployed with the Enterprise premise or outdoor for Smart City or Smart Transport.

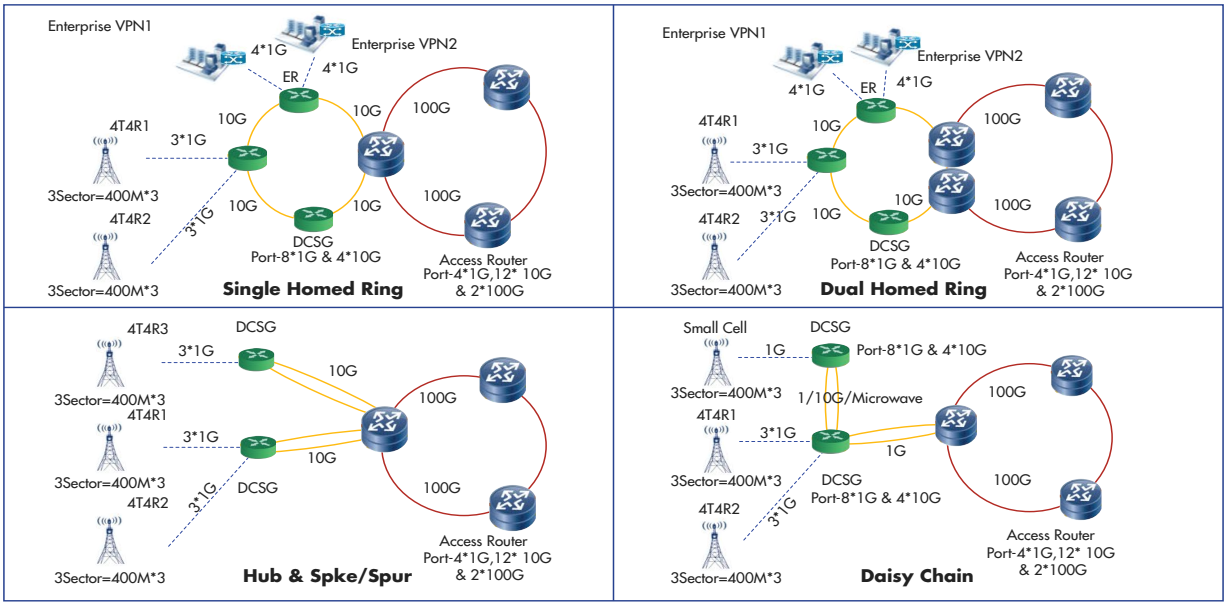


Figure 3: *DCSG and ER deployment topology*

2.3.2 DCSG Alternative Deployment Scenarios

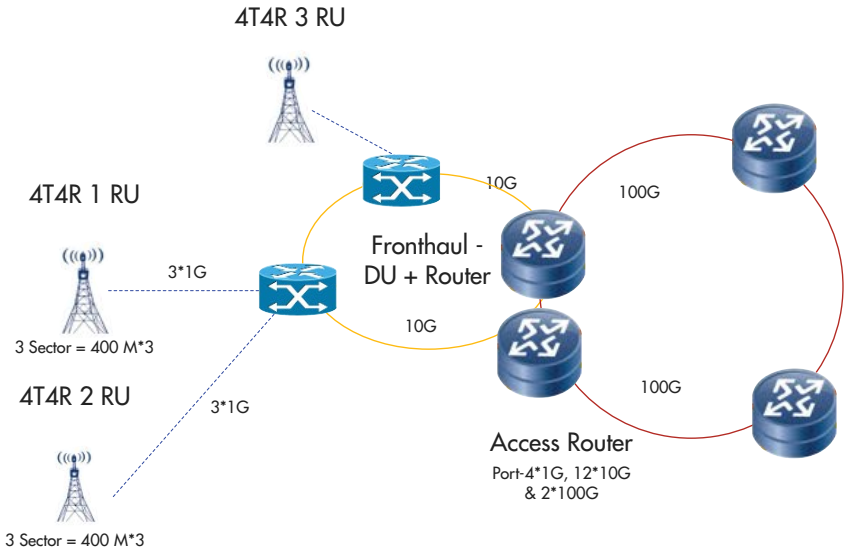


Figure 4: *DU + Router deployment*

In the ORAN architecture, the 5G 7.2 Split Option using F1 front haul divides the radio processing into 3 parts – Radio Unit (RU), Distributed Unit (DU) and Centralized Unit (CU). To reduce the number of components near the cell-site, there is interest to create a new network node that combines the DU processing and DCSG Routing:

- DU - High-PHY + RLC + MAC processing
- DCSG - MPLS-VPN Routing & Forwarding

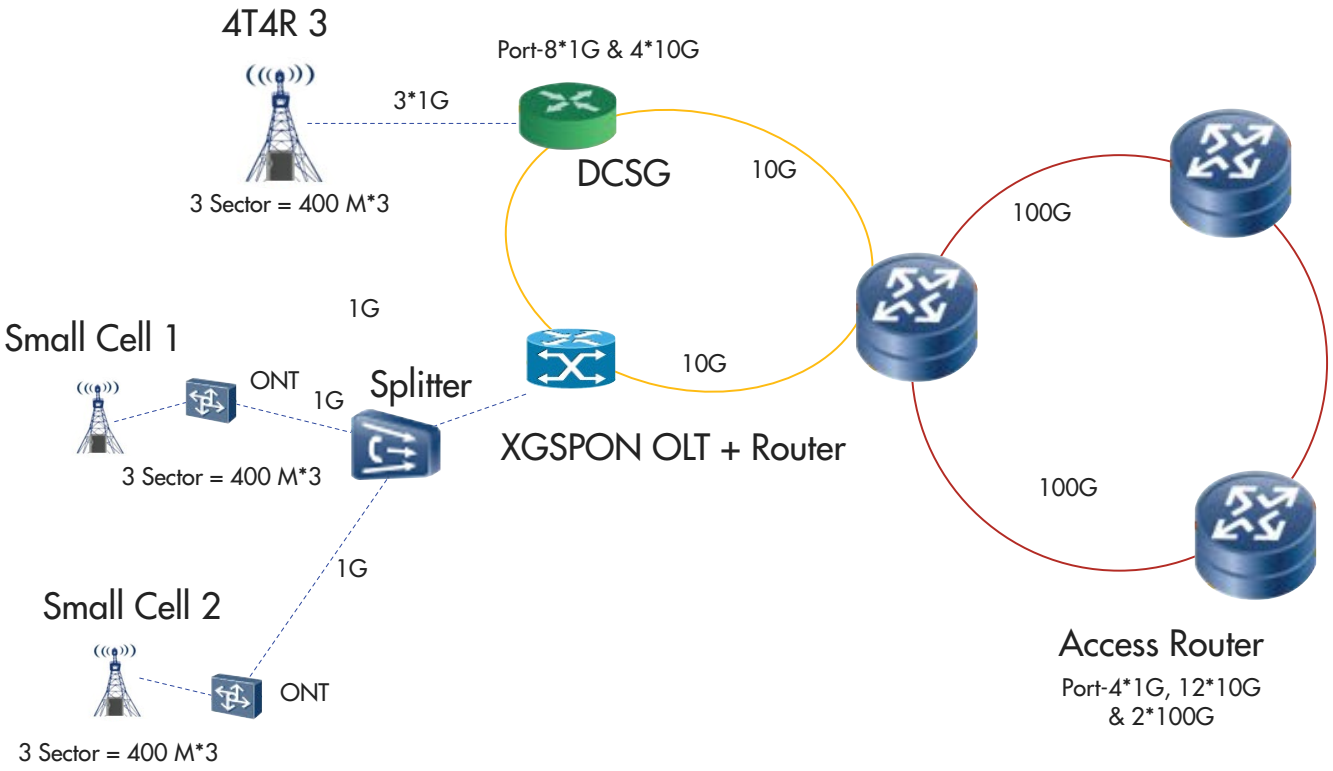


Figure 5: XGSPON + Router deployment

With small cells deployed in densely populated urban pockets, there can be an alternate XGSPON + Router deployment option wherever there is fibre connectivity. The XGSPON would be connected to each small cell using ONT. The Routing features would be similar to the DCSG.

2.3.3 SDN Controller Deployment Scenarios

The SDN Controller centrally manages and provisions all the network nodes within the Telecom operator's network. There can be 2 types of deployment option:

- Hierarchical SDN Controller (Option 1) – This is recommended by Telecom Infra Project (TIP). There are multiple SDN controllers per domain like Optical, IP, Microwave etc. The EMS is also hierarchical.
- Single SDN Controller (Option 2) – Some Operators are using a single SDN Controller to manage all the domains – Optical, IP, Microwave etc. The EMS is hierarchical and controlled per domain.

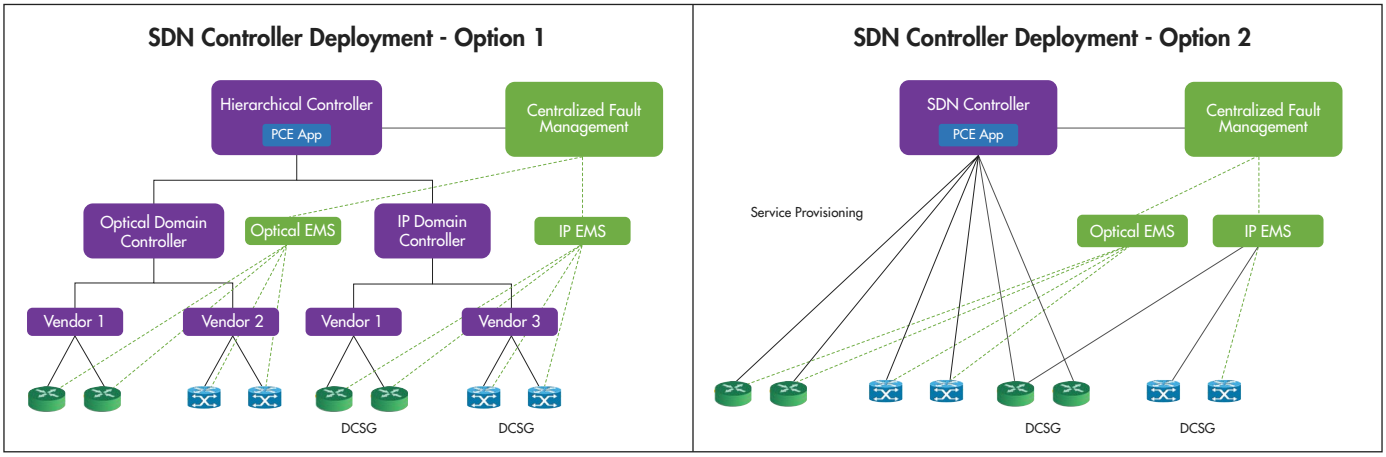


Figure 6: SDN Controller deployment Option

3 - Architecture

3.1 System Architecture

3.1.1 Disaggregated Architecture

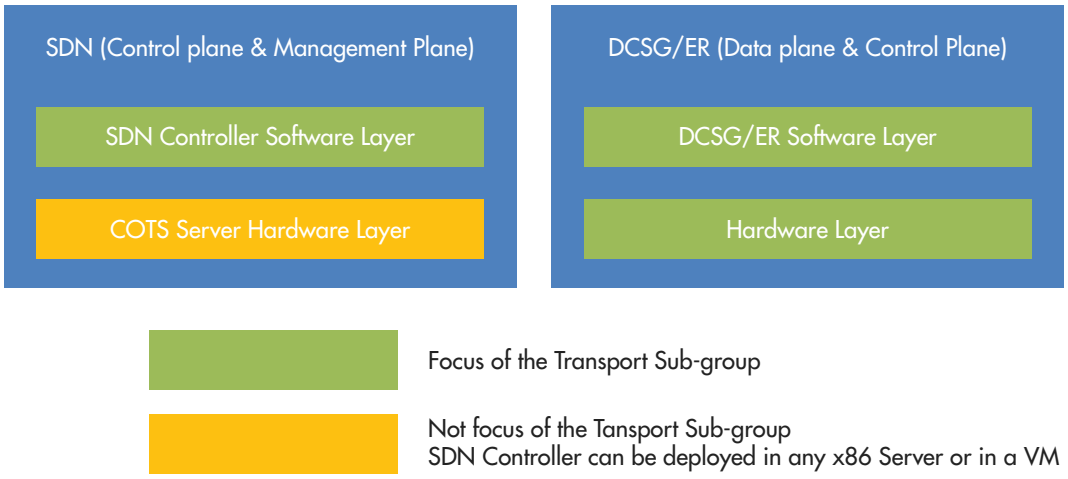


Figure 7: DCSG/ER and SDN Controller Disaggregation

Disaggregation is a key requirement for 5G deployment where the Software from any vendor can be integrated to the Hardware from any vendor. As part of the design of the DCSG and ER, a key requirement is that the data plane and control plane software of the DCSG or ER should work with White Box hardware of any ODM. The SDN Controller software application that is inherently disaggregated and runs on any COTS server. So, the Transport sub-team work is focused on these:

- Open-Source Software for DCSG
- Open HW Design for DCSG
- Open-Source Software for ER
- Open HW design for ER
- Open-Source Software for the SDN Controller

3.1.1 Disaggregated Architecture

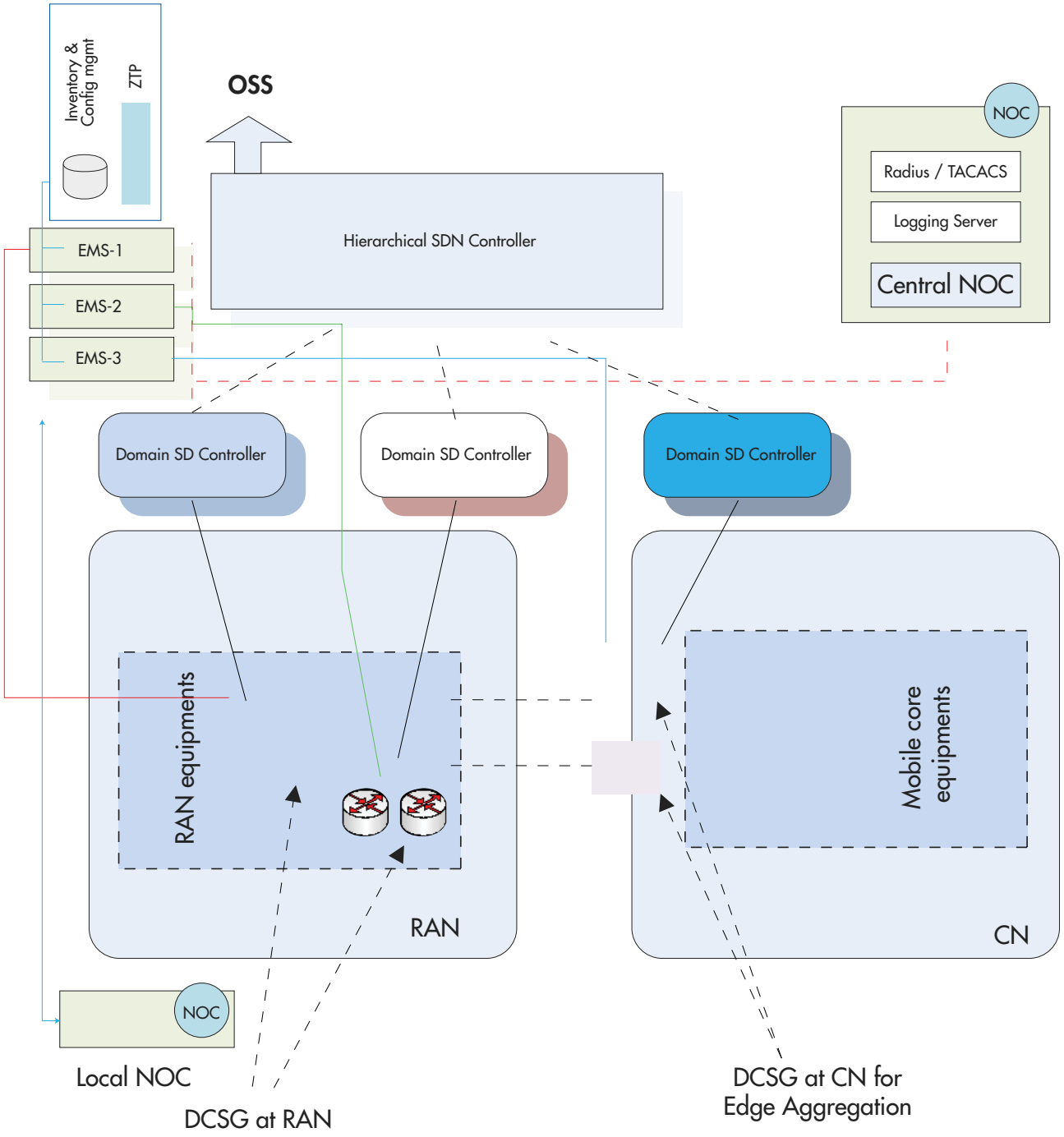


Figure 8: *IP Transport Component Interactions*

The DCSG is at least expected to interact with the following key Telecom infrastructure functions:

- **Domain SDN Controller:** The Domain SDN controller is expected to control, manage and configure services on a specific set of DCSG nodes, which share equal control characteristics either due to role/proximity/feature availability/vendor restrictions. The Domain SDN controllers are also expected to retrieve and be completely service aware of the capabilities of the DCSG nodes. The DCSG devices are expected to be fully service compliant to the Domain SDN controllers.
- **Hierarchical SDN Controller:** The Hierarchical SDN controller is responsible primarily for end-to-end service configuration of the Operator's entire Network Infrastructure. It is expected to input service definitions in a vendor agnostic manner and convert them to protocol specific or vendor specific translations, which should include the Domain SDN controller. DCSG nodes and their capabilities are not directly involved with Hierarchical SDN controllers. However, it should be able to provide service and operational primitives to the Domain SDN controller which can in turn make it part of the complete end-to-end service layout.
- **EMS:** Element Management System (EMS) is mandated again to be associated with the DCSG again in a group or individual subject to role/proximity/vendor restrictions.

The EMS component is required, as it can directly manage, monitor the DCSG at various stages;

a. During first time provisioning – the EMS system is expected to provide service to a ZTP system or manual provisioning system for configuration of the DCSG systems.
b. Inventory & configuration management – the EMS system discovery and accounting of the nodes will be used in maintaining the inventory management of the devices in the service providers' network.
c. Domain level operations and service monitoring – the EMS system will also be used for monitoring DCSG node level, alarms, faults and also to health check domain level service checks, for example, by enabling IP-SLA, Y.1731, CFM at local domain.

The EMS system provides the background infrastructure to provide a local NOC level dashboard.

The DCSG node are expected to provide the following Service and Operations interfaces:

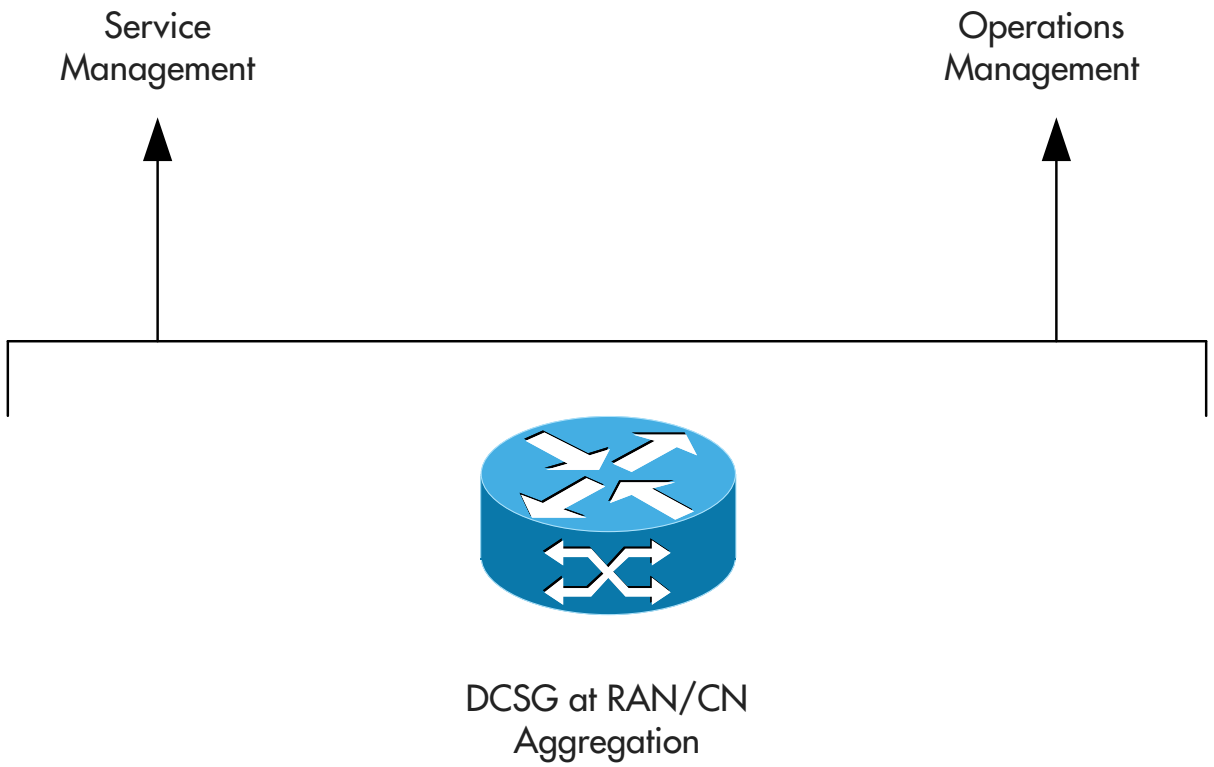


Figure 9: DCSG Management Interface

a. Service Management:

Services	Suggested mandated Control Interfaces/ Protocol	Optional Control Interfaces/Protocols	DCSG placement	Comments
Device discovery & ZTP	LLDP, Netconf, ONIE, ZTP DHCP	CDP	(All)	1 U
Configuration & software image management	TFTP, SCP, FTP Netconf with config rollback	(vendor specific)		300 mm
IPv4 Transport	IGP, BGP, Static		(All)	Yes
IPv6 Transport	IGP, BGP, Static		(All)	60 Gbps / 32 Gbps
Tunnelling	MPLS, SR, SRv6	IP/GRE	(All)	Yes
Alarm & Fault Indication	a. Service Alarms, Traps. b. System Alarms, Traps		(All)	Yes
OAM	OAM services over supported transport		(All)	Yes

b. Operations Management:

Services	Suggested mandated Control Interfaces/ Protocol	Optional Control Interfaces/Protocols	DCSG placement	Comments
Device control and configuration	CLI, Netconf	SNMP, Openconfig	(All)	
Device Alarm and Fault Indication	Device Alarms, Traps, Link & Transceiver Alarms, Traps		(All)	
Protocol level detailed statistics	CLI, Netconf, Telemetry	Openconfig	(All)	
Hardware level detailed statistics	CLI, Netconf, Telemetry	Openconfig	(All)	
Authentication and Authorization capabilities	RADIUS, TACACS, Syslog		(All)	
Active device & flow monitoring	RMON, SPAN, RSPAN, NetFlow, sFlow, Legal Intercept framework		(All)	

3.2 Hardware Architecture

3.2.1 Components of Hardware

The components of DCSG/ER involve data plane processing and control plane processing elements. The ER uses a single-chip processor for both control plane and data plane processing. DCSG uses ASIC for data plane processing and processor for control plane processing. ASIC is needed for DCSG as the data plane requirement is higher than 20 Gbps switching capacity. Processor can have multiple cores with the required supporting memory.

The power supply module can be AC, DC, AC+DC, redundant/non-redundant. The DCSG will have a redundant power supply option while ER will have a single power supply module.

Operation and management require FPGA/CPLD, forced cooling will require fan modules and DCSG also needs additional timing modules.

3.2.1 Components of Hardware

3.2.2.1 DCSG Hardware Architecture

The below figure explains the DCSG hardware architecture:

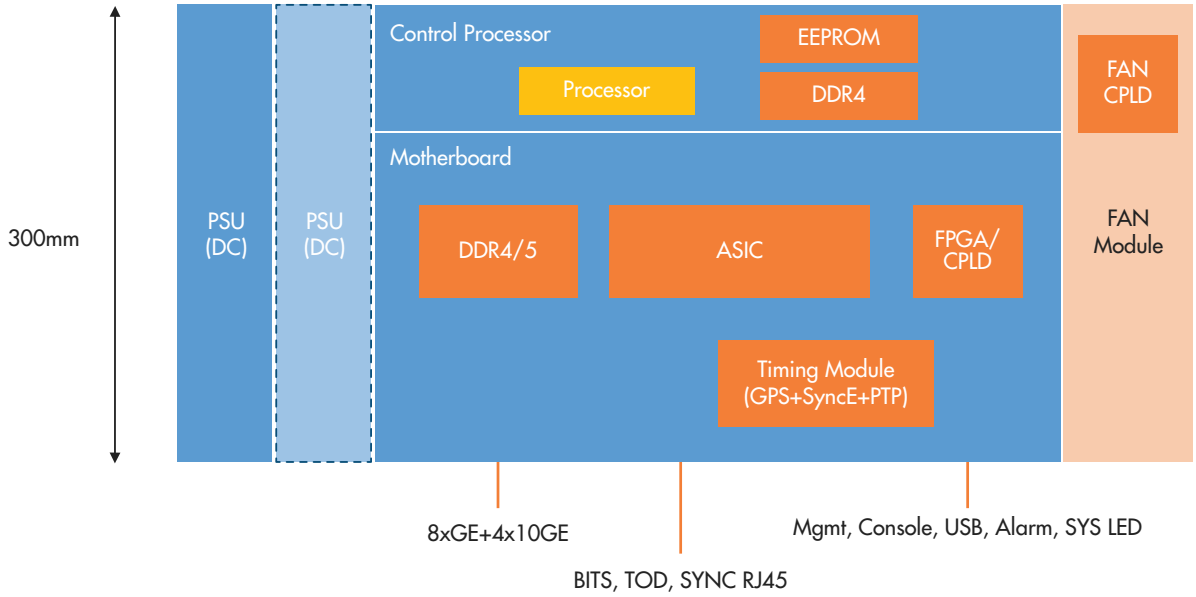


Figure 10: DCSG HW design

Depending on the switching capacity, we suggest two options for 5G DCSG:

- Option 1: 180Gbps (with SRv6)
- Option 2: 60/200 Gbps (No SRv6 but SR-MPLS)

The study above is based on ASICs from one vendor. Additional ASIC options are FFS.

3.2.2.2 ER Hardware architecture

The below figure explains the ER hardware architecture:

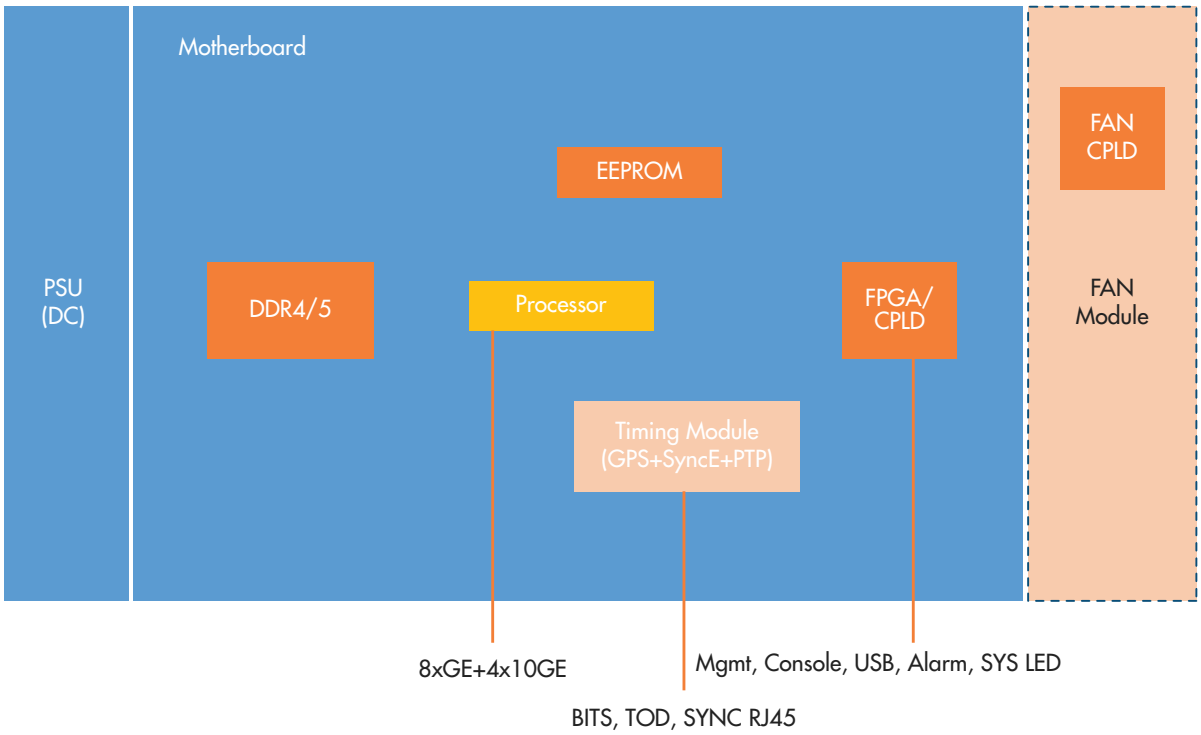


Figure 11: ER HW design

The options for this can be based on different processors with 8 or higher number of cores. We initially did the study based on x86. This is feasible for upto 60 Gbps with 16 x86 cores. ARM based processor is FFS.

3.3 Software

3.3.1 DCSG Design

The DCSG is deployed near the cell site, so there is a possibility of deployment of 100 of thousands of DCSG per Tier-1 operator. The DCSG Software has 3 important components:

- Management Plane
- Control Plane
- Data Plane

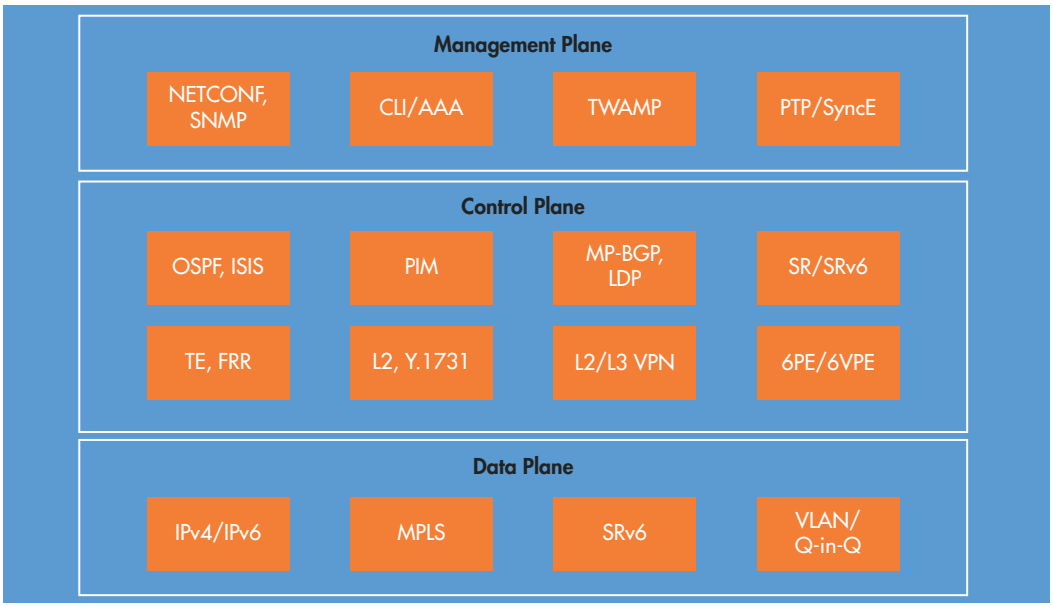


Figure 12: DCSG SW design

The Management Plane is used to configure, provision, debug and manage the DCSG. It is connected to the SDN Controller, EMS server for centralized management. It uses NETCONF, SNMP protocol to communicate with SDN Controller and EMS Server respectively. The Management processing is done with the host processor of the DCSG. Its main components include the CLI, Authentication, TWAMP, Zero Touch Provisioning, Telemetry collection and Timing solution - PTP, SyncE.

The Control Plane runs all the IGP, BGP and Multicast Routing Protocols. It also runs the LDP and Traffic Engineering related protocols for MPLS, Segment Routing and Layer 2 and Layer 3 VPN. The control plane also runs Layer2 protocols like Y.1731, LACP etc. The control plane processing is CPU intensive and runs on the host processor of the DCSG. The number of routes and labels is stored in the Routing Information Base (RIB) and Label Information Base (LIB). The size of the RIB and LIB defines the scalability of the DCSG.

The Data Plane does the packet processing of IPv4, IPv6, MPLS, SRv6 or L2 packets. 5G applications are guaranteed to have a latency of less than 10ms, so the DCSG should have minimum packet processing latency. Packet processing can be done in the ASIC or NPU to reduce latency of the DCSG. The forwarding table for routes and labels is stored in the Forwarding Information Base (FIB) of the data plane.

3.3.1 DCSG Design

The ER is a trimmed down version of DCSG deployed in the Enterprise and Smart City/Smart Transport that does not need Segment Routing, Traffic Engineering and Time Synchronization Capabilities. The ER Software has 3 important components:

- Management Plane
- Control Plane
- Data Plane

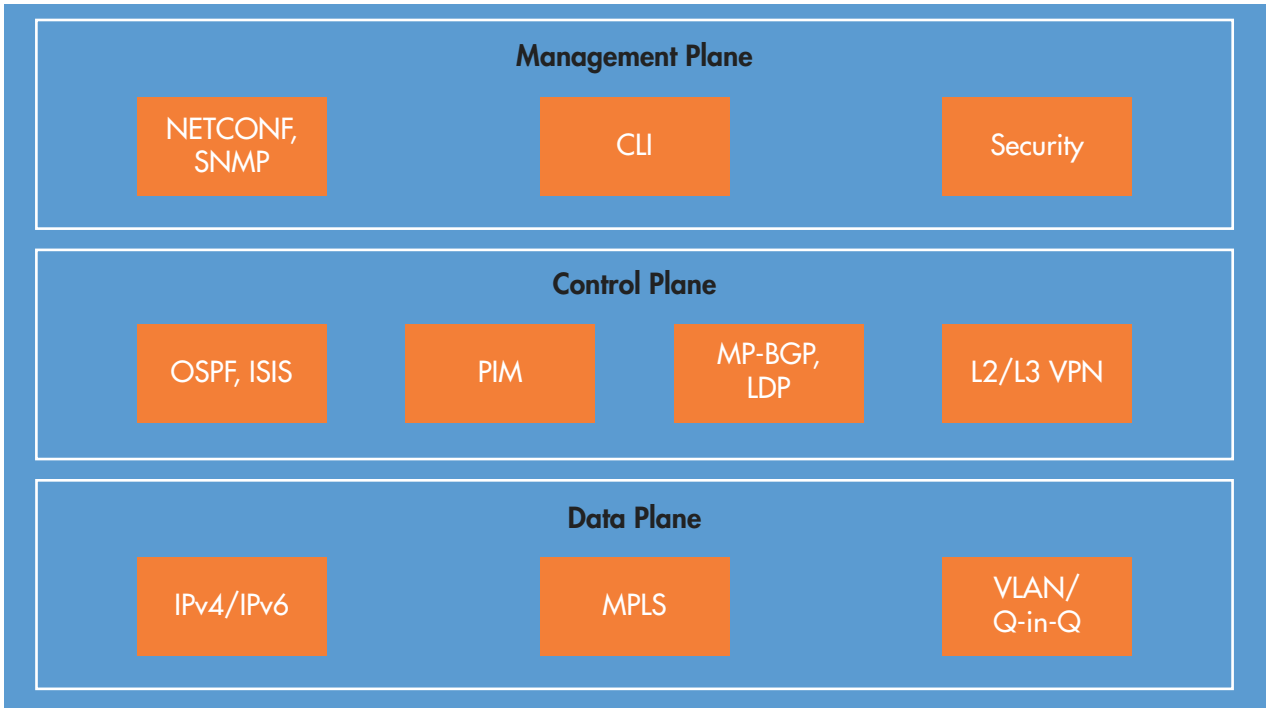


Figure 13: ER SW design

3.3.3 SDN Controller Design

The management plane is expected to follow a Hierarchical architecture where the top level Hierarchical SDN controller is expected to interface with the end-to-end service orchestration as part of a network slice creation. The Hierarchical SDN controller will further trigger the domain specific SDN controllers for the actual provisioning of the DCSG/routers in terms of their configurations or paths.

For provisioning to work via SDN controllers, the first step is to discover these DCSG in the network. For this, if the DCSG has a network connectivity with the domain SDN controllers, it can directly use the NB or SB interface (depending on the protocol chosen) of these SDN controllers to add its device level configuration (IP address/port info for south bound side connectivity).

The SDN controller can then trigger a SB protocol connection with the DCSG. If the DCSG has no knowledge of the SDN controller, a network management entity (orchestrator – Day 0 configuration) will be required to register the DCSG components with the domain SDN controllers. Once the DCSG components are connected with domain SDN controllers, they are ready for provisioning.

There are two aspects to the device level provisioning at DCSG:

- Path creation across network

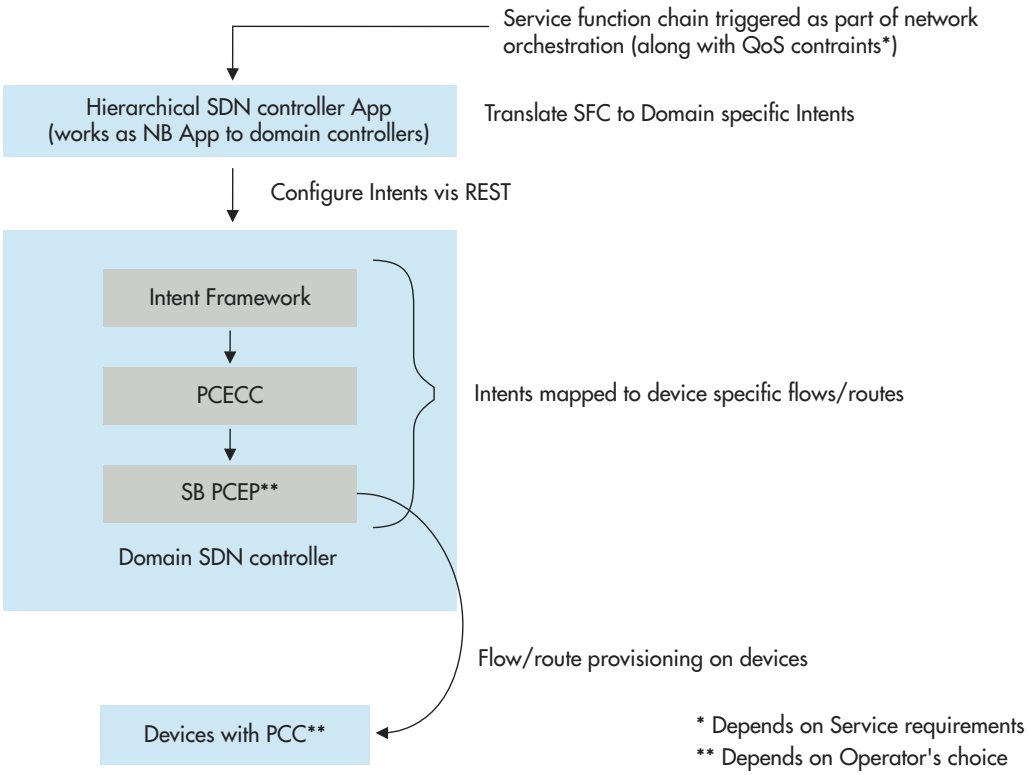


Figure 14: SDN Controller Framework

The end-to-end orchestrator is expected to provide the desired network path in terms of connectivity and QoS/QoE configuration to the Hierarchical SDN controller. The Hierarchical SDN controller acts as a northbound SDN application to the domain controllers. It is responsible for segregating the end-to-end service level configuration to domain specific service level configuration. Further, it will translate the domain specific service level configuration into domain specific intents which will be fed to the domain controllers via the REST interface between the Hierarchical SDN controller and the domain controllers.

Domain specific intents define the routing path between domain specific network entities (DCSG, routers) in terms of desired connectivity, with the required SLA. The intent framework of the domain SDN controller will translate these intents into paths, using the network topology learned via the SB interface and the performance criteria/constraints defined by the domain specific intents. For topology formation, the domain SDN controllers will leverage SB protocols like BGP-LS and act as a BGP speaker. Based on the link, state information received in the BGP-LS packets, the SDN controller will create a topology and use that along with the domain intent information to create a path between two endpoints.

The domain SDN controller will further convert these paths into specific flows/routes to be configured on the DCSG/routers using SB protocols (PCEP, NetConf, OpenFlow etc).

- Day1/Day2 Device level configurations

The Hierarchical SDN architecture can also be leveraged by the end-to-end orchestrator to pass some Day1/Day2 configurations to the DCSG components. This will require a northbound application (device config app) at the domain SDN controller which opens us a REST interface to receive device configurations in JSON/XML format. The domain SDN controller will also require the specific YANG models supporting these device configurations. The device config app will use these YANG models to generate Java objects based on the JSON/XML content received from the northbound side. These Java objects will be used to create RPC contents which the domain SDN controller will trigger towards the managed DCSG components using a southbound side protocol level connection (most commonly uses NetConf).

4 - Evaluation

Various Open-Source routing stacks were considered for DCSG and ER. Routing Stacks includes Bird, Zebra, Quagga, FRRouting and DANOS. The Evaluation Criteria was:

- IGP protocol Support
- BGP and Multicast Support
- MPLS, SR Support
- L2 and L3 VPN Support
- Traffic Engineering Support
- Integration with NETCONF, SNMP, Telemetry for northbound
- OS and HW Platform Support
- Integration to Test Automation
- Active Contribution from various entities and lively mailing-list, slack etc.

After initial analysis, DANOS and FRRouting were considered for evaluation of DCSG and ER.
For the domain SDN controllers, two most popular Open-Source SDN controllers were evaluated – Open Networking foundation's ONOS (Open Network Operating system) and Linux foundation's ODL (OpenDaylight). Following are some of the criteria based on which, this evaluation was done:

- Feasibility to use the intent frameworks of these controllers to realize the 5G SLA requirements.
- Authentication/authorization support.
- Protocol support on the northbound and the southbound interface of the controllers.

- Support for high availability in the s/w architecture of these controllers.
- Long term release plans from the developer community.
- Liveliness on the developer/user mailing lists for diagnostic support and other discussions.
- Integration into wider systems by close interactions with other Open-Source initiatives across industry.

4.1 Open-Source Evaluation

4.1.1 Routing Stack Evaluation for DCSG and ER

4.1.1.1 System architecture supported by Open-Source

FRRouting

FRRouting release 7.5.1 is a Routing Stack application that has the advantage of being ported into multiple run-time environments like VNF, CNF or Linux. It creates multiple processes to ensure scalability on multi-core processors. FRRouting has its own IPC to communicate between its processes. It has a GPLv2 license.

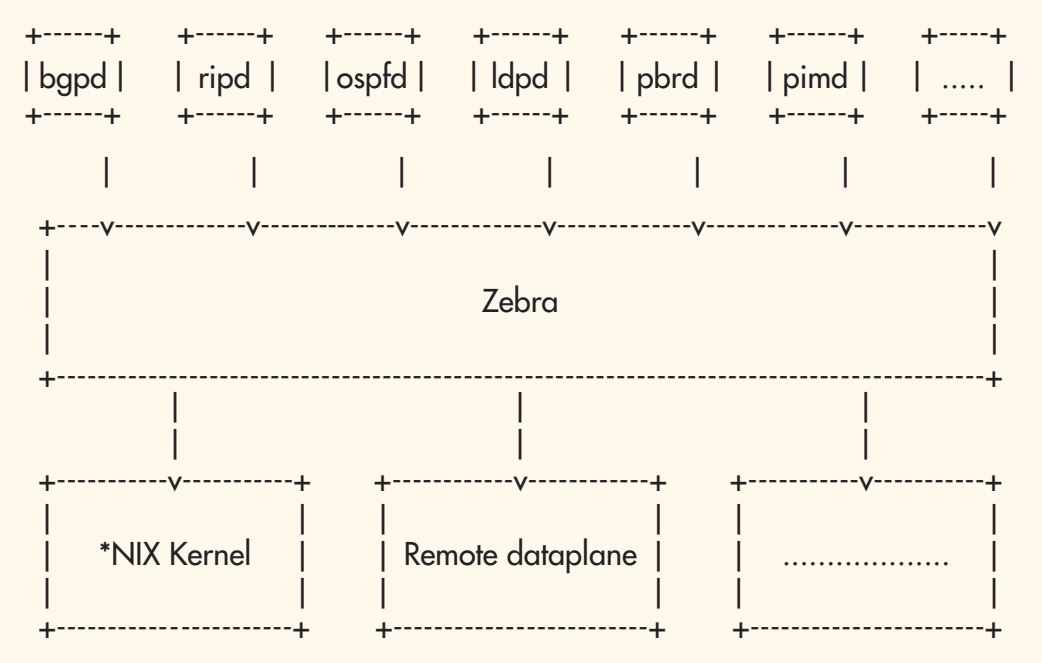


Figure 15: FRRouting Architecture

FRR is a suite of daemons that work together to build the routing table. Each major protocol is implemented in its own daemon and these daemons talk to a middleman daemon (zebra), which is responsible for coordinating routing decisions and talking to the data plane.

This architecture allows for high resiliency, since an error, crash or exploit in one protocol daemon will generally not affect the others. It is also flexible and extensible since the modularity makes it easy to implement new protocols and tie them into the suite. Additionally, each daemon implements a plugin system allowing new functionality to be loaded at runtime.

All of the FRR daemons can be managed through a single integrated user interface shell called vtysh which connects to each daemon through a UNIX domain socket and then works as a proxy for user input. In addition to a unified front end, vtysh also provides the ability to configure all the daemons using a single configuration file through the integrated configuration mode. This avoids the overhead of maintaining a separate configuration file for each daemon. FRR is currently implementing a new internal configuration system based on YANG data models.

DANOS

DANOS release 2105 is a Routing Operating System that uses FRRouting for the Routing Stack and has a management platform and data plane ported to ASIC that supports around 200 Gbps of switching traffic. DANOS has integrated multiple Open-Source software like FRRouting, nDPI, DPDK, OpenNsl etc. Thus, DANOS has multiple licenses LGPL, GPLv2 etc. DANOS Architecture has the management plane, control plane and the data plane.

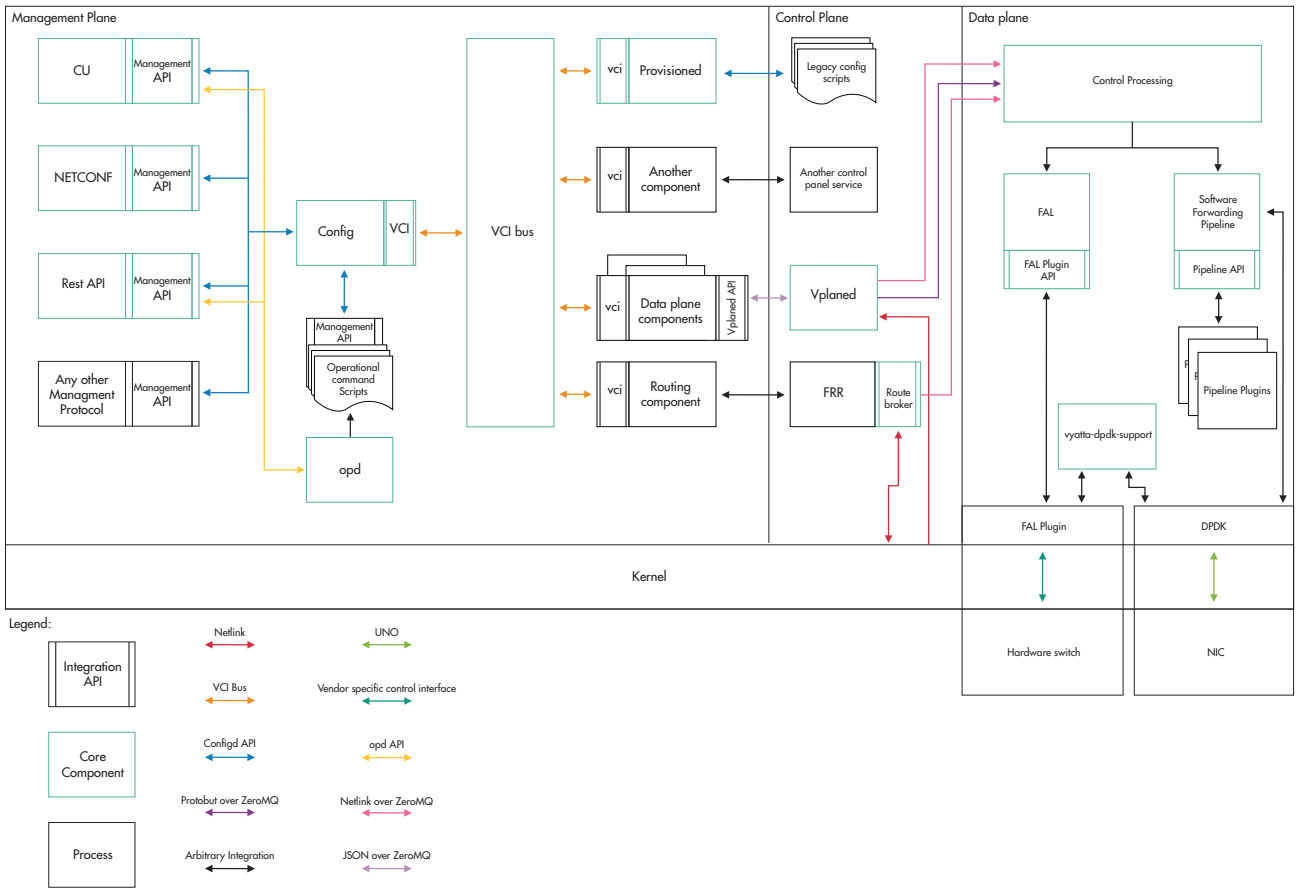


Figure 16: DANOS Architecture

The management plane consists of the configuration and operational infrastructure. Extensions that are built on these layers include the CLI, RestAPI, NETCONF daemon, all of our feature integration components and any other management scripts. Configd is a daemon that manages access to the system's desired configuration, operational data and RPCs. OPD is a daemon that creates modelled operational commands. The VCI Bus is a message bus that is used for communications between VCI components.

The route broker is a daemon that ensures that eventual consistency is achieved between the control plane, kernel, and data plane FIB. Vplaned is a daemon that manages interaction with the data plane. ZMQ is used as a transport to send all of this information to the data plane.

The data plane is a combination of the DPDK based software forwarding pipeline and the FAL interface to hardware forwarding pipelines. It provides the necessary mechanisms for bridging the DPDK pipeline with switching Silicon based on user configuration.

4.1.1.2 Configuration Overview

DANOS has comprehensive support for NETCONF/YANG, FRRouting has partial support for NETCONF/YANG. Both DANOS and FRRouting have CLI support.

4.1.1.3 Feature Set Supported

FRRouting has comprehensive support for Routing and MPLS protocol. There is a Telecom Specific Technology group within FRRouting to develop solutions for Telecom use cases.

DANOS has good support for Northbound management and White box HW integration. DANOS uses FRRouting for the Routing and MPLS Stack, but the MPLS feature integration is not complete in DANOS.

4.1.1.4 Hardware requirements by Open-Source

FRRouting Stack runs on x86, ARM processor, VM and Docker.

DANOS runs on x86 processor and is also ported on White box with NPU/ASIC. DANOS also runs on VMs. It is not ported to an ARM processor or Docker yet

4.1.1.5 Product support across industry

FRRouting project has existed for more than a decade and is used in commercial deployments by companies like Cumulus Networks, Volta Networks. There has been active contribution from Telecom Operators like Orange into FRRouting.

DANOS is relatively new and has been seed contributed by AT&T. Most features in DANOS are from an Open-Source project called Vyatta that had multiple commercial deployments.

4.1.2 SDN Controller Evaluation

4.1.2.1 System Architecture Supported by Open-Source

Both ONOS and ODL support intent-based path formation. Intents can be leveraged by the Hierarchical SDN controller to realize the path formation as per the SLAs of the 5G network slice the operator is trying to create.

ONOS supports an intent framework which can be used to feed the desired contents via ONOS CLI or via RESTful interface. The intents are compiled into installable intents based on state of the network (learnt via Southbound protocols like BGP-LS) and the behaviour requested by the northbound side intent. The northbound side intents are configured in terms of paths between managed network entities – point to point, point to multipoint, multi-point to single point and host to host intents. The installable intents are translated into flow level configurations which can be sent to the managed devices using southbound side protocol (NETCONF, OpenFlow etc.) connections between the controller and the devices.

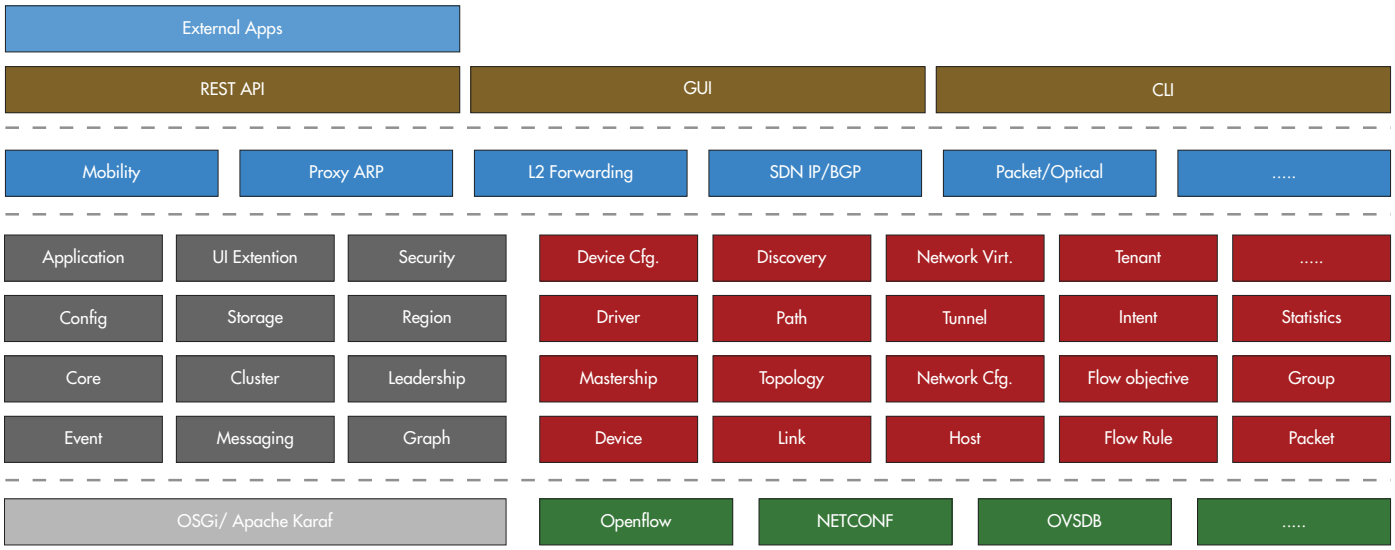


Figure 17: ONOS Architecture

ODL also supports intent-based service management which can be used to feed the desired contents via REST or CLI. The ODL intents are defined in terms of subjects (endpoints involved), action (block/allow), constraints (exclude/include) and conditions (recurring/scheduled). These intents are passed to an intent renderer which, based on the network state and the configured intents, can create the detailed flow level configuration and send it to the managed devices using southbound side protocol (NETCONF, PCEP, OpenFlow etc.) connections between the controller and the devices.

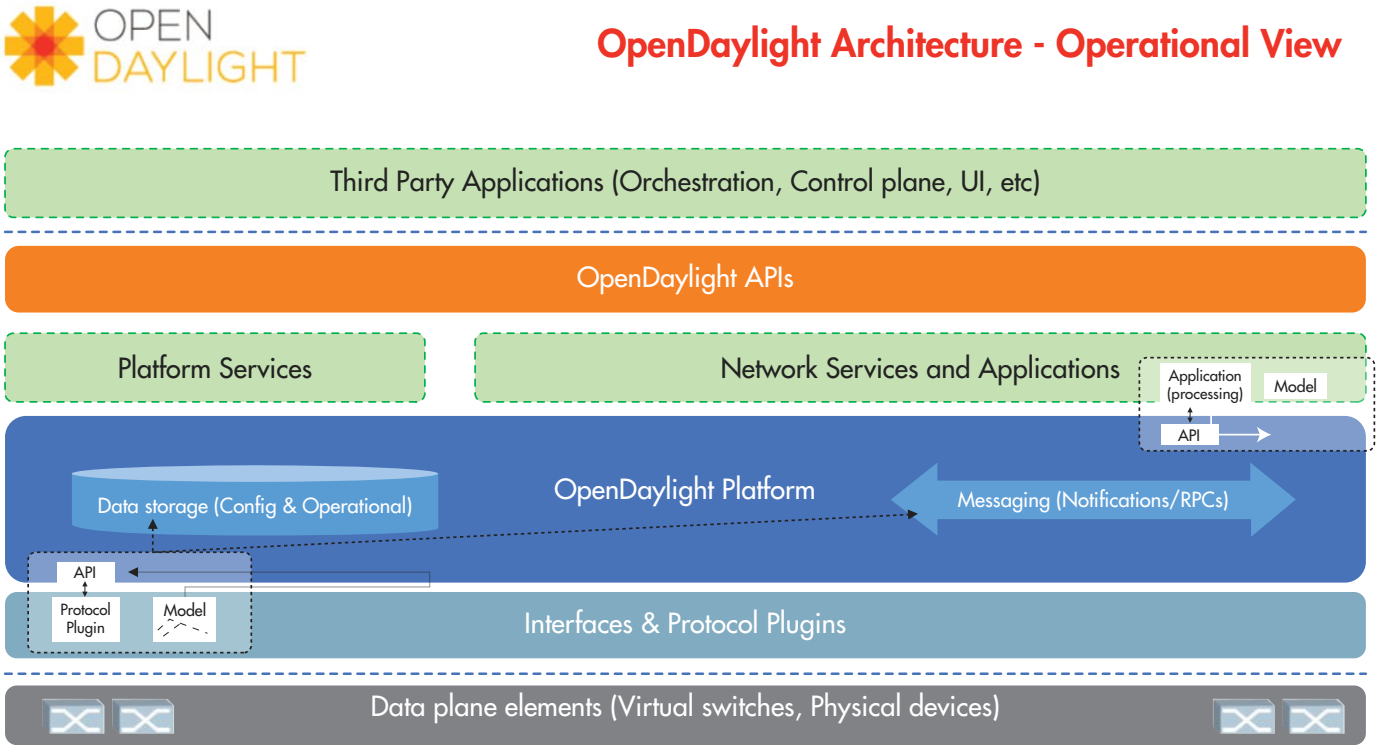


Figure 18: ODL Architecture

When configuring intents, SDN controllers will need to handle conflict resolution – how to handle the scenarios when operators configure more than one intent and these multiple intents are contradicting. Both ONOS and ODL might be supporting conflict resolution at intent level, but only the ODL documentation states it very clearly as part of its intent framework.

4.1.2.2 Configuration Overview

Configuration of the domain SDN controllers involves two aspects – configuration of the northbound applications with business logic and configuration of the network (device discovery in terms of managed devices and links).

For the ONOS SDN controller, the northbound application is either included in the ONOS package before deployment or these apps can be installed later on a running cluster in form of ONOS Application archives (OARs). This involves copying the application on the target machine running ONOS and activating it. Also, doing so one of the ONOS instances is enough – it gets replicated on the rest of ONOS instances of the cluster itself.

The device discovery of the network is basically adding the managed devices to ONOS. This is achieved via a network configuration service in ONOS which accepts all the device specific information in a JSON format.

This JSON contains the list of devices including their connectivity information (management IP and port on which ONOS can make a SB connection) and the SB protocol the device supports. The network configuration service of ONOS further triggers the SB drivers to create a SB connection with these devices. This configuration JSON can also be used to declare the links of these devices for the scenarios where the links cannot be discovered by any IGP/BGP protocols of the SDN controller.

In ODL, there are two options for a new user/custom application – either integrate it with ODL using the model-driven SAL (MD-SAL) archetype or develop external apps and use the RESTCONF to communicate with ODL. Generally, ODL applications are northbound plugins which use OSGi interfaces. The plugin defines a list of services, which it provides and consumes. For the list of consumed services, all necessary dependency structure is resolved by OSGi using the definitions in the pom.xml file.

For device discovery in ODL, taking the NETCONF example, a new NETCONF device can be registered with the SDN controller using the concept of NETCONF-connectors. A new NETCONF connector for a NETCONF device is created by triggering a POST/PUT request to RESTCONF layer of ODL – this PUT/POST request has a JSON/XML attached to it which includes the connectivity information (management IP and port on which ONOS can make a SB connection). This creates a MD-SAL mount point for the device in ODL which can be used to allow applications and remote users (over RESTCONF) to interact with the mounted/connected devices.

4.1.2.3 Feature Set Supported

Protocol support

At the protocol level, both ODL and ONOS have similar capabilities. Both the SDN controllers support REST and RESTCONF at the northbound side. At the southbound side, following is the list of protocols supported by each of them:

- ODL: OpenFlow (1.0, 1.3). BGP, PCEP, SNMP, NETCONF, LISP, CAPWAP, OCP, OPFLEX, SXP, USC, SNBI, LACP
- ONOS: Openflow, netconf, gnmi, bgp, grpc, gnoi, lisp, ovsdb, p4runtime, SNMP, tl1, xmpp

Authentication and authorization

ODL supports the AAA framework and has support for role-based access control. ONOS capabilities are very limited in this area, with ONOS supporting only authentication (basic authentication, public/private key authentication).

High availability

Both ONOS and ODL support multi-instance cluster, which is needed for a high availability. If one instance of ONOS/ODL fails, the managed devices connected to that instance are moved to some other instance of ONOS/ODL. But the clustering and storage mechanism of ODL is inbuilt – it has an embedded raft mechanism for all storage – this might be simpler, but a catastrophic failure of the complete cluster means data loss. Also, ODL only supports strict consistent data replication which means that no transaction can complete till majority of the cluster nodes confirm that they have successfully persisted the data. This can have major performance impacts if the data is large or frequent.

ONOS, on the other hand, has a clear separation of storage and control. It uses a separate Open- Source s/w Atomix for all storage/cluster management. Also, ONOS supports two models for data replication – eventually consistent and strongly consistent. Using the eventually consistent model can show better performance in case the system can tolerate some level of local data loss.

ODL inherently supports geo-clustering where different instances of a cluster can be placed in different geographical locations (with latency impacts though) or different data centres. ONOS documentation, on the other hand, talks of an inter-cluster concept (project ICONA) – where multiple ONOS clusters might be placed in a geographically distributed manner.

4.1.2.4 Hardware requirements by Open-Source

Both ODL and ONOS have similar minimum hardware requirement:

- CPU: 2 Cores; RAM: 2 GB; Storage: 10-20 GB

For better performance, the above-mentioned configuration can be enhanced in terms of cores and memory.

4.1.2.5 Product Support Across Industry

ODL has been a popular SDN controller choice in a data centre kind of environment. It offers active integration with VIM frameworks like OpenStack, OPNFV and Kubernetes. One of the most popular orchestration frameworks used by operators across the globe, ONAP by Linux foundation, also uses ODL as the network controller to instantiate the network configuration workflow across its VNFs.

ONOS meanwhile has strong support from ONF. ONF has been actively working on SD-RAN architecture using micro-services based ONOS SDN controllers for implementing the near real- time RIC in the Open RAN area.

Both ONOS and ODL have an active developer and user community giving LTS releases every 6-8 months

4.2 Current Limitations

ONOS supported PCEP prior to the ONOS 2.5 release, but this protocol support has been removed in the 2.5 release.

FRRouting has limitations in Traffic Engineering, EVPN, SRv6 and NETCONF Support. The major limitation of DANOS is the MPLS Support and MPLS related features.

4.3 Gap Analysis

4.3.1 DCSG/ER Routing Stack Analysis

Features	FRRouting (Release 7.5.1, License GPLv2)	DANOS + FRRouting (DANOS 2105, License LGPL, GPLv2, ...)
IPv4, IPv6 Support	Yes	Yes
OSPF, ISIS, BGP, PIM	Yes	Partial
MPLS, SR Support	Yes	No
SRv6 Support	No	No
L2 and L3 VPN	Yes	No
6PE, 6VPE	No	No
Traffic Engineering, SR- TE, PCE	Partial	No
Layer 2 - Y.1731	No	No
Timing Protocol – PTP, SyncE	No	No
NETCONF	Partial	Yes
CLI and Management	Partial	Yes
Test Automation and CI	Yes	No
Platform Support	Docker, VM, Linux	VM, Linux, Broadcom - Ufispace S9500-30XS and Edgecore AS5916-54XKS
Active Contributors	Cumulus Networks, VMware, Orange, netdef, LabN, 6Wind, Volta, Niral Networks	AT&T, Niral Networks

4.3.2 SDN Controller Analysis

Features	OpenDaylight (ODL) Release 13 (Aluminium), Eclipse Pub License 1.0	Open Network Operating System (ONOS) + Atomix
Types of Intents available and conflict resolution	Configured via REST or CLI; In terms of subjects (endpoints), action, constraints and conditions; Supports conflict resolution among intents as per official documentation	Configured via REST or CLI; In terms of paths - P2P, P2M, M2P, host-to-host Conflict resolution mentioned in plans but not in any official documentation
Authentication/ authorization support	Supports AAA RBAC also present	Supports Authentication (basic authentication, public/private key authentication.
NB/SB protocol options	NB: REST/RESTCONF SB: OpenFlow (1.0, 1.3). BGP, PCEP, SNMP, NetConf, LISP, CAPWAP, OCP, OPFLEX, SXP, USC, SNBI, LACP	NB: REST/RESTCONF SB: Openflow, NETCONF, gnmi, bgp, grpc, gnoi, lisp, ovsdb, p4runtime, SNMP, tl1, xmpp
Community activeness	Active on user/developer community	Active on user/developer community
	2 releases per year	LTS release every 6—8 months
HA architecture	Supports internally managed multi-instance cluster Supports Geo-distributed Active/ Backup Setup (with latency impacts)	Supports multi-instance cluster with cluster management via Atomix. Supports multiple ONOS clusters to share information about their networks - ICONA
Collaboration with other Open-Source communities	Active integrations with OpenStack, OPNFV, ONAP, Kubernetes	Active work on SD- RAN

- For a single pane of glass management, operators will have a strong requirement to integrate the SDN controller information with their existing network management systems. This requires more exploration around the new OSS/BSS systems based on the TM forum standards, their Open-Source alternatives and their integration with the SDN controllers for a consolidated FCAPS.
- There is a wide-spread study around the emerging configuration/management protocols like GNMI, GNOI. It requires more study in terms of leveraging these protocols to replace legacy SNMP/NETCONF management and their support/stability in the SB side of the SDN controllers.
- For vendor neutrality, multiple vendors are extending the support of OpenConfig models in their new platforms. Eventually, device management should move to these new OpenConfig models and their integration with the SDN controllers.

4.4 Roadmap

FRRouting is working on multiple Telecom domain features that includes:

- Traffic Engineering - Implementation of Path Computation Element Protocol (PCEP) in FRR to achieve PCC feature suitable to control RSVP-TE tunnel and Segment Routing Path and Path Daemon to control Segment Routing path.
- BGP-LS - BGP Link State (RFC7752) implementation in FRR
- Segment Routing – Completion of Segment Routing for both OSPF end IS-IS
- Northbound Layer – NETCONF and YANG models for the all the FRRouting protocols

DANOS is also working on Telecom related features that includes:

- Support for new White Box Hardware Platform
- Northbound Management using NETCONF
- MPLS Support

5 - Conclusion

FRRouting is working on multiple Telecom domain features that includes:

5.1 Recommended Open-Source/HW

	Open-Source Project	White box HW Option
DCSG	DANOS 2105	ASIC based switch/router (non-redundant)
Edge Router	FRRouting 7.5.1	X86 based vCPE/uCPE
SDN Controller	ODL Aluminium Better Security, well defined conflict resolution, Similar protocol support (compared to ONOS)	x86 server or VM

FRRouting is the recommended Open-Source software for ER.

DANOS is the recommended Open-Source software for DCSG.

For the SDN controller, it is recommended to use ODL for the following reasons:

- ODL has support for AAA and role-based access control.
- ODL inherently supports conflict resolution across the defined intents.
- ODL supports all the desired protocols at the NB and SB side of the domain SDN controller.

5.2 Minimum Viable Product (Can contain the PoC details)

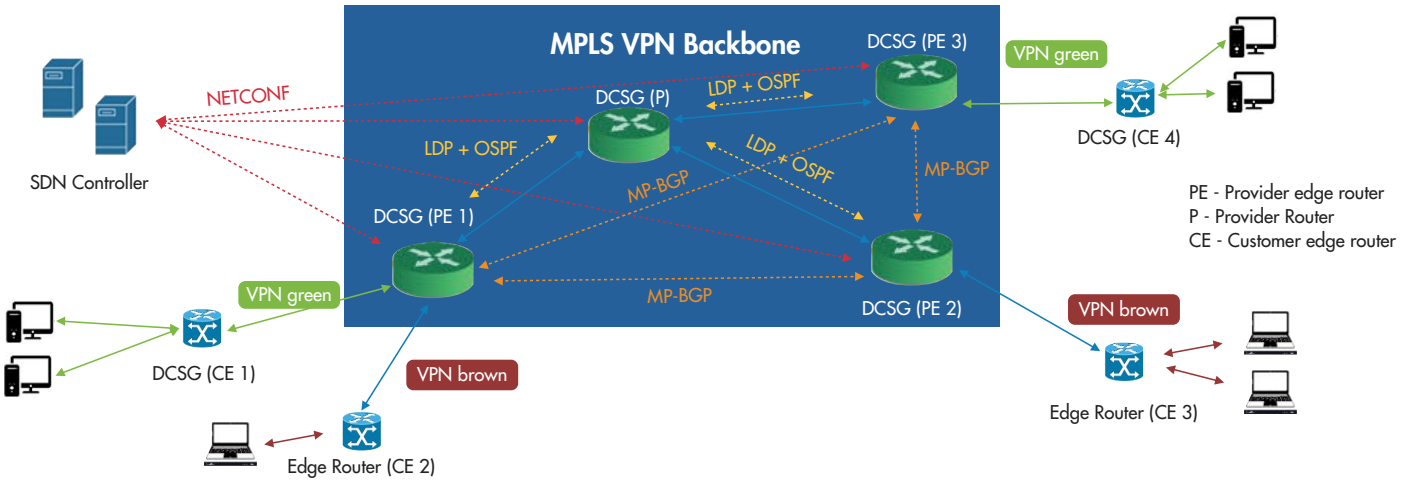


Figure 18: ODL Architecture

The POC setup contains the DCSG, Edge Router and the SDN Controller. LDP, OSPF and MP-BGP are configured to create the L3 VPN or L2 VPN backbone. The DCSG acts as Provider Edge and Provider router for the MPLS backbone, the ER acts as the Customer Edge Router within the Enterprise. There can be multiple Virtual Private Networks for each Enterprise. The SDN Controller is deployed centrally and manages all the DCSGs.

Hardware Recommendation for PoC

- ER - CPU: 8 Cores; RAM: 8 GB; Storage: 64 GB
- DCSG - ASIC: 60/180 Gbps switching capacity, 8 Cores processor with RAM of 8 GB
- SDN Controller - CPU: 8 Cores; RAM: 8 GB; Storage: 64 GB

5.3 Areas of Future work

To move towards Minimum Viable Product (MVP) for POC and subsequent deployment, the following are the list of work items required:

	Open-Source Project	Roadmap Work Item
DCSG	DANOS 2105	1. Porting all the MPLS, L2 VPN, L3 VPN from latest of FRRouting 2. Porting all the SR, Traffic Engineering, PCEP features from latest of FRRouting 3. Porting all the BGP, PIM, OSPF, ISIS features from latest of FRRouting 4. Addition of Timing and L2, Y.1731, 6PE, 6VPE, BGP-LS, SRv6, EVPN support 5. Addition of Test Automation, CI Support
Edge Router	FRRouting 7.5.1	1. Addition of basic L2 and Security support 2. Completion of the NETCONF support for all protocols 3. Integration to Intel DPDK or ARM acceleration for the forwarding at higher throughput
SDN Controller	ODL Aluminium	1. Developing northbound app acting as a Hierarchical SDN controller. Integrate it with the orchestrator 2. Design intents as per operator's SLA requirements 3. Integrate YANG models, translation to valid XMLs and pass to NETCONF SB. 4. Integration with existing OSS/BSS/NMS for FCAPS 5. Telemetry support to be added for network analytics 6. Create an upgrade strategy to move to LTS releases periodically.

6 - References

- https://telecominfraproject.com/
- https://www.o-ran.org/
- https://github.com/danos
- https://danosproject.atlassian.net/wiki/spaces/DAN/pages/684457996/Architectural+ Overview
- https://github.com/frrouting
- http://docs.frrouting.org/en/latest/overview.html
- https://github.com/opendaylight
- https://www.opendaylight.org/about/platform-overview
- https://opennetworking.org/onos/
- https://www.opencompute.org/

Applications Sub-Team

1 - Introduction

This addendum is the report of the Application Sub-Team within the Task-Force and supports the "Feasibility of Open-Source for 5G- Final Overall Report" issued by the TSDSI Task-Force on Open-Source for 5G. Please read the main section of the Report to get the context, background and overall recommendations of the Task Force.

Intelligence is swiftly becoming a necessity for the deployment, optimization and operation of Radio network due to increasing complexity. Open RAN brings in concept of AI/ML enabled RAN intelligent controller (RIC) aiming to enhance network performance by enabling third party applications through open APIs and closed loop control RAN Intelligent Controller (RIC). This document captures the feasibility of Open-Source usage in the development and deployment of RIC.

O-RAN reference architecture specifies the introduction of a hierarchical RAN Intelligent Controller (RIC) platform and leverages "compute" capabilities of a cloud-native environment to enable AI/ML driven intelligent decisions and automation in the RAN. Figure 1 pictorially shows the overall structure of the O-RAN logical architecture.

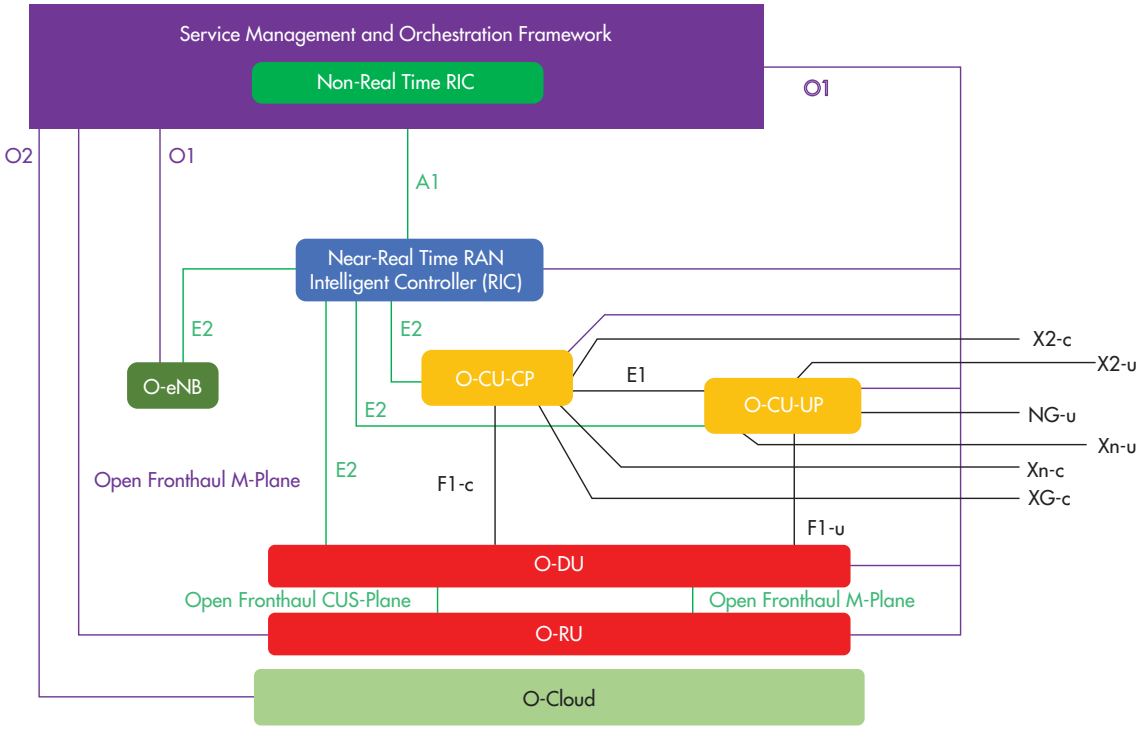


Figure 18: ODL Architecture

The Non-RT control functionality (> 1s) and near- Real Time (near-RT) control functions (< 1s) are decoupled in the RIC. Non-RT functions include service, configuration and policy management, RAN analytics and model-training for the near-RT RAN functionality. Some of the trained models and real-time control functions produced in the non- RT RIC are distributed to the near-RT RIC over A1 interface for runtime execution. rApps are applications designed to run on the Non-RT RIC. Such modular application leverages the functionality exposed by the Non-RT RIC to provide value added services with respect to intelligent RAN optimization and operation. Near-RT RIC enables near real-time control and optimization of O-RAN (O- CU and O-DU) nodes and resources over the E2 interface with near real-time control loops (i.e., 10ms to 1s). The Near-RT RIC uses monitor, suspend/stop, override and/or control primitives to control the behaviours of O-RAN nodes. The near-RT RIC hosts xApps that use E2 interface to collect near real-time RAN information to provide value added services using these primitives, guided by the policies and the enrichment data provided by the A1 interface from the Non-RT RIC.

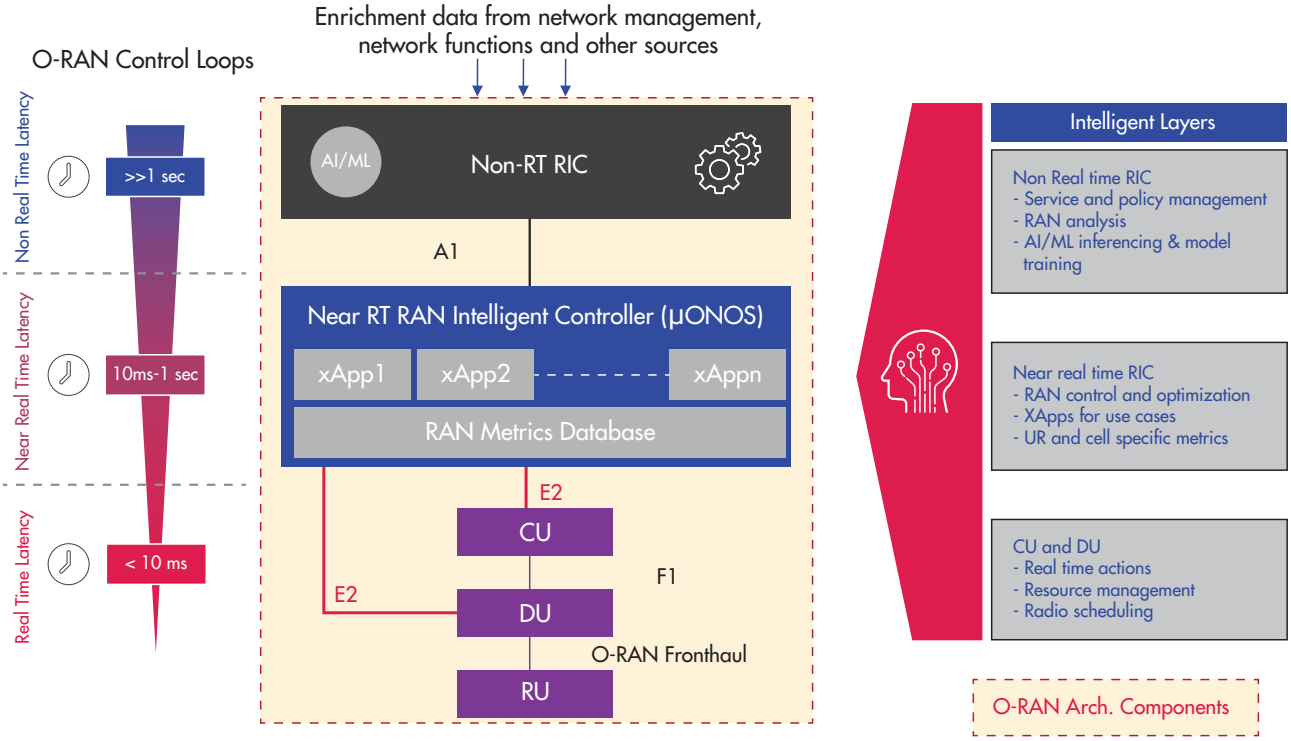


Figure 18: ODL Architecture

1.1 Scope

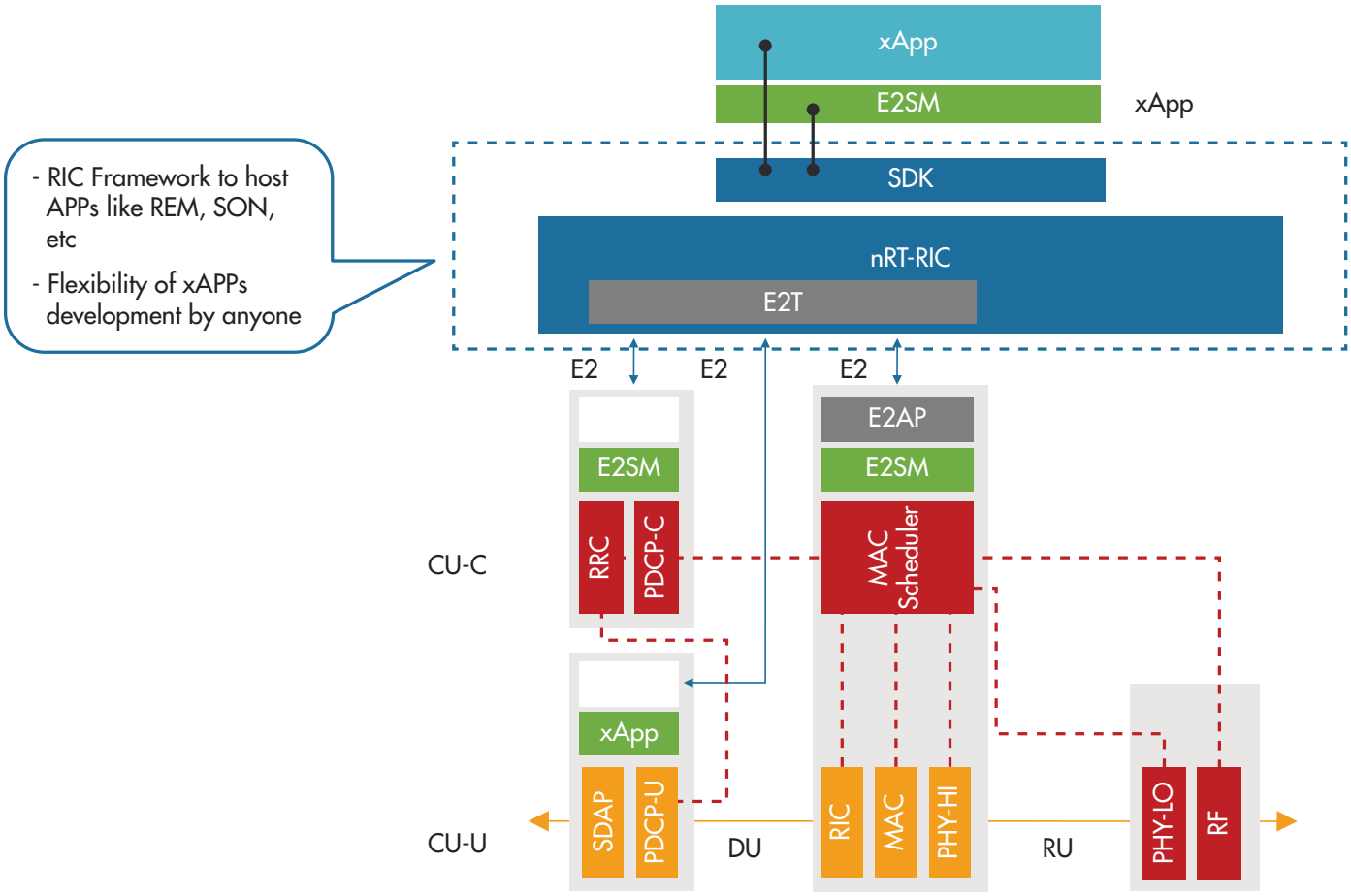


Figure 3: RIC Framework

In the current study, the focus is on the RIC framework. RIC framework will provide the platform to host the APPs to optimize the RAN functionalities like RRM, SON etc. RIC framework will also provide the open APIs for the APP development.

1.2 Evaluation Criteria

Below table describes the criteria that are considered for the evaluation of Open-Source components available in this domain. Section 4 captures further details about the evaluation.

Criteria	Details
O-RAN compatibility	Adherence to the O-RAN ecosystem
Full Software Availability	Access to full software for users to enhance and contribute.
Software Quality	Software quality checks as per standard industry practices. Having software analysis to maintain commercial grade software
Interoperability	Interoperability with 3rd party Components (Commercial and Test equipment)
Hardware Requirement	Any specific hardware requirement to run the software
Deployment Considerations	Experiences of the ecosystem to adapt the Open-Source in various deployments
Roadmap for O-RAN standards	Organization with clear path for further software upgrades as per O-RAN standards
Automation and CI	Industry standard practices for CI and Automation.
Community Support	Maturity and support from community. Experience of renowned Open-Source component initiatives

1.3 Document Organization

This document is organized as follows:

- Section 2 describes the requirement specifications for near-RT and non-RT RIC. It also covers various use cases that are defined by different consortia.
- Section 3 introduces the overall architecture for near-RT and non-RT RIC components.
- Section 4 covers the evaluation of SD-RAN and ORAN-SC RIC frameworks based on the evaluation criteria described in above section. Details of features, hardware requirement and roadmaps are also provided.
- Finally, in Section 5, conclusion is drawn on the feasible Open-Source option. This section also covers the aspects of creating a minimum viable product based on Open-Source RIC platform and suggestions on possible areas of future work.

2 - RIC Requirement Specification

2.1 RIC Generic Requirements

- Support for all Open Interfaces to/from Near RT and Non RT RIC

- Support for O1, A1, E2 and other interfaces that are part of O-RAN standard
- RIC architecture supporting O-RAN Standards
- Support Open APIs for 3rd Party xAPPs and rApps
- Support evolution of network management including predictive and self-learning solutions
- Support enhancement of framework to include India specific use cases

2.2 Near-RT RIC Functional Requirements

The Near-RT RIC architecture shall follow the Near-RT RIC requirements specified in document O-RAN.WG3.RICARCH and the following requirements:

- The near-RT RIC shall consist of multiple xApps and a set of framework functions that are commonly used to support the specific functions hosted by xApps.
- The near-RT RIC shall provide an open framework to allow on-boarding and Life Cycle Management (LCM) of xApps to support Near-RT RIC related use cases.

2.2.1 Framework Requirements

- The near-RT RIC shall provide a database function that stores the configurations relating to E2 nodes, Cells, Bearers, Flows, UEs and the mappings between them.
- The near-RT RIC shall provide ML tools that support data pipelining.
- The near-RT RIC shall provide a messaging infrastructure.
- The near-RT RIC shall provide logging, tracing and metrics collection from Near-RT RIC framework and xApps to SMO.
- The near-RT RIC shall provide security functions.
- The near-RT RIC shall support conflict resolution to resolve the potential conflicts or overlaps which may be caused by the requests from xApps.

2.2.2 xApps Requirements

- xApps may enhance the RRM capabilities of Near-RT RIC.
- xApps shall provide logging, tracing and metrics collection to the Near-RT RIC.
- xApps shall provide a descriptor that will include basic information (configuration, metrics and control) about the xApp.
- xApp descriptor components shall include the following:

- Configuration: The xApp configuration specification shall include a data dictionary for the configuration data, i.e., meta data such as a YANG definition or a list of configuration parameters and their semantics. Additionally, it may include an initial configuration of xApps.
- Control: xApp controls specification shall include the types of data it consumes and provides that enable control capabilities (e.g., xApp URL, parameters, input/output type).
- Metrics: The xApp metrics specification shall include a list of metrics (e.g., metric name, type, unit and semantics) provided by the xApp.
- The xApp descriptor shall also provide the necessary data to enable their management and orchestration.

2.2.3 Open API Requirements

- Near-RT RIC shall provide an open API enabling the hosting of 3rd party xApps and xApps from the Near-RT RIC platform vendor.
- Near-RT RIC shall provide an open API decoupled from specific implementation solutions, including a Shared Data Layer (SDL) that works as an overlay for underlying databases and enables simplified data access.

Note: Communication between xApps is for further study.

2.3 Near-RT RIC Protocol and Interface Requirements

2.3.1 E2 Termination

- E2 Termination terminates the SCTP connection from each E2 Node.
- E2 Termination routes messages from the xApps through the SCTP connection to the E2 Node.
- E2 Termination decodes the payload of an incoming ASN.1 message enough to determine message type.
- E2 Termination handles incoming E2 messages related to E2 connectivity.
- E2 Termination receives and respond to the E2 Setup Request from the E2 Node.
- E2 Termination notifies xApps of the list of RAN functions supported by an E2 Node based on information derived from E2 Setup and RIC Service Update procedures.
- E2 Termination notifies the newly connected E2 Node of the list of accepted functions.

2.3.2 A1 Termination

A1 termination provides a generic API for the Near-RT RIC by means of which it can receive and send messages via A1 interface. These include, e.g., A1 policies and enrichment information received from the non-RT RIC, or A1 policy feedback sent towards the non-RT RIC.

2.3.3 O1 Termination

O1 termination communicates with SMO via O1 interface and exposes O1-related management services from Near-RT RIC.

- O1 termination exposes provisioning management services from Near-RT RIC to O1 provisioning management service consumer.
- O1 termination supports managing xApps via NETCONF.
- O1 termination supports translation of NETCONF to Near-RT RIC internal APIs.
- O1 termination exposes FM services to report faults and events from Near-RT RIC to O1 Fault Management (FM) service consumer.
- O1 termination exposes PM services to report bulk and real-time PM data from Near- RT RIC to O1 Performance Management (PM) service consumer.
- O1 termination exposes file management services to download ML files, software files, etc. and upload log/trace files.
- O1 termination exposes communication surveillance services to O1 communication surveillance service consumer.

2.3.4 Near-RT RIC security requirements:

One of the targets of security functions in near-RT RIC is to prevent malicious xApps from abusing radio network information (e.g., exporting to unauthorized external systems) and/or control capabilities over RAN functions. The security requirements for the 3GPP LTE eNB and 5G NR gNB follows 3GPP standards.

2.4 Non-RT RIC Functional Requirements

- Non-RT RIC shall support data retrieval and analysis; the data may include performance, configuration or other data related to the application (recommended data shown in required data section for different use cases).
- Non-RT RIC shall support relevant AI/ML model training based on the data mentioned above for non- real time optimization of configuration parameters in RAN or near-RT RIC, as applicable for the use case.

- Non-RT RIC shall support relevant AI/ML model training for generating/optimizing policies and intents to guide the behaviour of applications in near-RT RIC or RAN, as applicable for the use case
- Non-RT RIC shall support training of relevant AI/ML models to be deployed/updated in near-RT RIC as required by the applications.
- Non-RT RIC shall support performance monitoring and evaluation
- Non-RT RIC shall support a fallback mechanism to prevent drastic degradation/fluctuation of performance, e.g., to restore to the previous policy or configuration
- Non-RT RIC shall be able to produce enrichment information through data analysis
- Non-RT RIC shall not update the same policy or configuration parameter for a given near-RT RIC or RAN function more often than once per second
- Non-RT RIC shall be able to update policies in several near-RT RICs.
- Security requirements for non-RT RIC are FFS.

2.4.1 A1 Interface

The O-RAN architecture includes a non-Real Time (RT) RAN Intelligent Controller (RIC) and a near-RT RIC which are connected through the A1 interface.

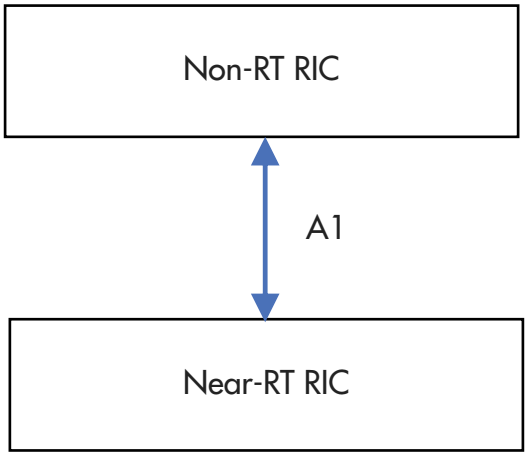


Figure 4: A1 Interface Between the Non-RT RIC and the Near-RT RIC

Based on high level directives (goals for the system) expressed in RAN Intent and on observables (events and counters) provided over O1, the non-RT RIC defines policies that are provided to the near-RT RIC over the A1 interface. The purpose of the A1 policies is to guide the RAN performance towards the overall goal expressed in RAN Intent. The A1 policies are declarative policies that contain statements on policy objectives and policy resources applicable to UEs and cells.

2.4.2 A1 Interface Requirements

- A1 interface shall support communication of policies from Non-RT RIC to Near-RT RIC.
- A1 interface shall support AI/ML model deployment and update from Non-RT RIC to Near-RT RIC.
- A1 interface shall support communication of enrichment information from Non-RT RIC to Near-RT RIC.
- A1 interface shall support feedback from near-RT RIC for monitoring AI/ML model performance.
- A1 interface shall support the policy feedback from Near-RT RIC to Non-RT RIC.

2.5 ORAN Control Loops

Near RT RIC control loop of < 1Sec
Non-RT RIC control loop of > 1Sec

2.6 Use Cases Defined by Consortia

2.6.1 ORAN WG1 Use Cases

O-RAN Use Cases	
QoS based Resource Optimization	- Radio resource allocation based on varying radio conditions. - To meet the requirements of reliability, latency and bandwidth simultaneously for diverse traffic situations. - To follow/satisfy the 3GPP defined QoS specifications. - Reallocate radio resources in case QoS requirements are not fulfilled.
QoE Optimization	User satisfaction-oriented analysis of the system performance. - QoE is translated into QoS work item - Two separate use cases with single work item - Two different call flows shall be decided.

O-RAN Use Cases	
Traffic Steering	Optimal load balancing through cell reselection and Handover decisions for the UE. - Analysis of load Calculations, hence Modifying cell priorities. - Predictive load balancing through AI/ML.
Massive MIMO Beamforming Optimization	Capacity Enhancement is obtained - by beamforming of the transmitted signals - by spatially multiplexing data streams for both single user (SU) and for multi user (MU) MIMO. Beamforming increases received signal power while decreasing interference generated by other users, hence resulting into higher SINR and enhanced user throughput.
RAN Slice SLA Assurance	Guaranteed Service Level Agreements along with RAN Slicing. RAN Slicing: End-to-end connectivity through control strategies such as PRB isolation, prioritized MAC scheduling and BLER control. RAN Slice SLA Assurance: RAN Slicing strategies with support of high data rates, UE densities, service availability, low latency, etc.
Slice Subnet Management	Management aspects of network slicing: Network Slice Instances (NSI) and Network Slice Subnet Instances (NSSI) are defined by 3GPP. - NSI: End-to-end network slice - NSSI: A part of NSI (such as a RAN slice) When the RAN part consists of O-RAN network functions, management and provisioning of RAN NSSIs are needed to be handled based on O-RAN capabilities and architecture. For this purpose, this work item is needed to enable O-RAN slice subnet instance provisioning and management as part of end-to-end 5G network slice delivery including SMO, Non-RT RIC, Near RT RIC, A1, O1, E2 interfaces aspects.

(Note: All the proposed/discussed use cases in the O-RAN community are listed here as a ready reference for readers.)

2.6.2 TIP Use Cases

O-RAN Use Cases	
Massive MIMO Beamforming Optimization	Traffic steering, an evolution of mobile load balancing, is a widely used network solution to achieve optimal traffic distribution based on desired objectives. Predictive load balancing will be achieved through AI/ML

O-RAN Use Cases	
Coverage and capacity optimization	Interference detection and mitigation. Address cell overshoot issues. Minimize coverage gaps
Inter-frequency and inter-RAT handover	Handovers controlled through the RIC
Accurate real time user location	Prediction of accurate real time user location through the information received from base stations

Use Cases	
Data energy optimization through power savings by load-adaptive mode	Power saving techniques based on the load in the system
SSB (Synchronization Signal Block) beamset optimization	Optimized SSB allocation based on user distribution, calculated using the number of detected UEs per SSB for each sweeping direction
Multi-user pairing	AI/ML assisted UE pairing while scheduling for MU-MIMO in massive MIMO system
Uplink channel estimation	Enhanced UL channel estimation through AI/ML techniques
Inter-cell interference coordination	Inter-Cell Interference (ICI) caused by the simultaneous usage of the same spectrum in different cells. Inter-Cell Interference Coordination (ICIC) at RIC can mitigate the impact of ICI on system performance.
Atmospheric ducting interference mitigation	In some weather conditions, in the Earth's atmosphere, low densities at higher altitudes causes reduced refractive index, which bends the signals back towards the Earth. So, in these scenarios, signals can propagate in the Atmospheric Duct i.e., Layer with Higher Refractive Index. Due to ducting phenomenon, in TDD systems, the downlink symbols of aggressor base station can travel longer distance and will interfere the uplink symbols of the Victim base station. This impacts coverage and capacity. Ducting mitigation techniques are deployed at RIC.

(Note: All the proposed/discussed use cases in the TIP community are listed here as a ready reference for readers.)

2.6.3 ETSI Use Cases

Use Cases	
SLA management	The specific parameters exposed for monitoring are agreed and specified between the RIC policies and the RAN KPIs. The specific application performance data to be collected could be a topic for future standardization.
Location-based service recommendation	The user service recommendation application can make use of machine learning and/or inference engine to determine proper services for users at the moment. The service recommendation can be done in cooperation with big data analysis which is fulfilled by backend server in internet.
Bandwidth allocation manager for applications	Different applications running in parallel on the same cell might require specific static/dynamic up/down bandwidth resources. In some cases, different sessions running in parallel in the same application can each have specific bandwidth requirements. As all these applications and application sessions are competing over the same shared bandwidth resources, it is suggested that a central bandwidth resource allocator exists as on RIC platform.
Optimizing QoE and resource utilization	Overall QoE perceived by the end users as well as utilization of the resources can be optimized with smart selection and combination of the paths used for the user plane.

2.6.4 NGMN Use Cases

Use Cases	
CoMP Evolution	Coordinated multipoint processing (CoMP) is regarded as an effective method to solve the problem of inter-cell interference and improve the cell-edge user throughput. With the CoMP technology, several neighbouring cells could be jointly processed or coordinated for cell- edge users in order to avoid interference and improve cell-edge users' throughput.
Verticals URLLC Use Cases	Enabling ultra-low latency and/or ultra-reliability, required by such use cases as industrial automation, tactile internet, augmented and virtual realities or autonomous automotive applications, has shown to involve important challenges. A flurry of activities has been ongoing in research, industry initiatives and standardization to analyse and address these challenges towards enablement of these use cases and transformation of relevant industries.

Use Cases	
Machine-to-Machine Communication	Traditionally M2M communication features low data-rate transmission. With emergence of more and more M2M applications, M2M services are presenting more and more characteristics and requirements in terms of bandwidth, power efficiency and speed and so on.

(Note: All the proposed/discussed use cases in the NGMN community are listed here as a ready reference for readers.)

2.7 Flow diagrams for Near-RT RIC

2.7.1 Report Service:

Near-RT RIC uses an RIC Subscription procedure to request that E2 Node send a REPORT message to Near-RT RIC and the associated procedure continues in E2 Node after each occurrence of a defined RIC Subscription procedure Event Trigger.

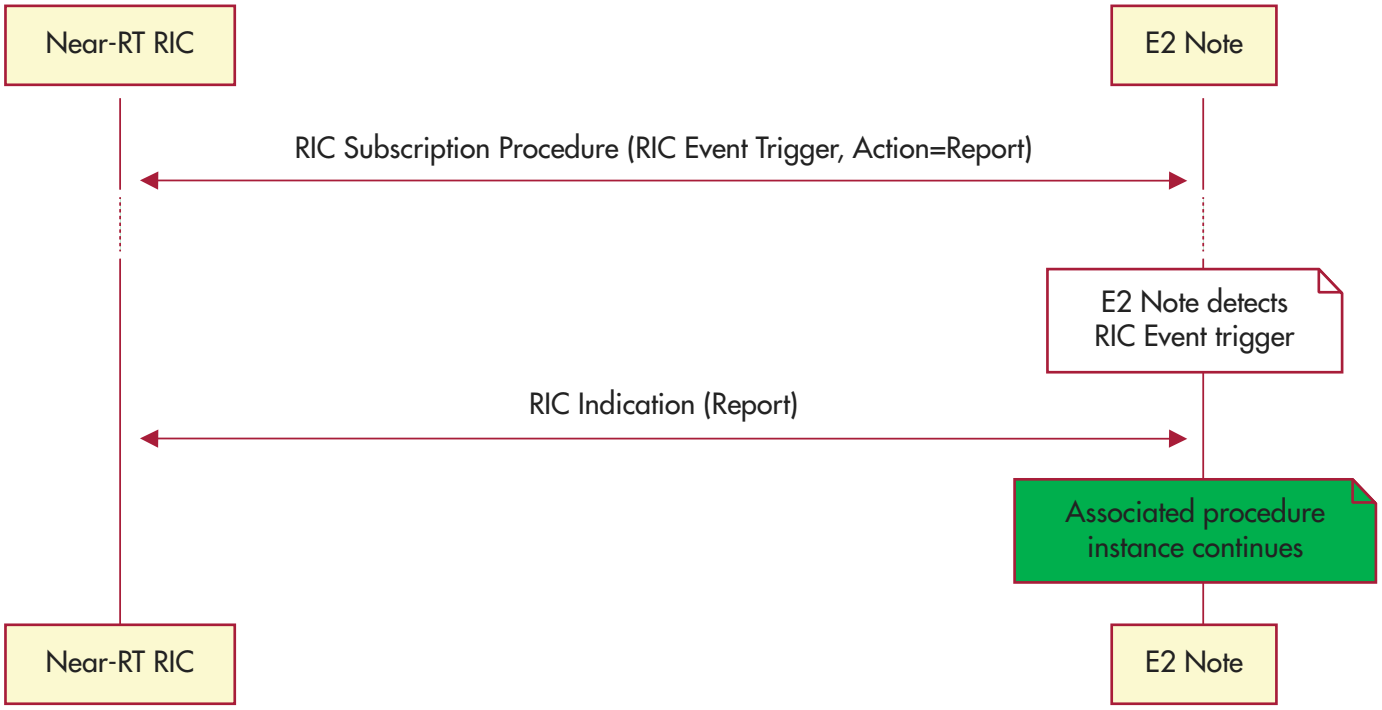


Figure 5: RIC Indication Procedure

2.7.2 Insert Service:

Near-RT RIC uses a RIC Subscription to request that E2 Node send an INSERT message to Near-RT RIC and suspend the associated procedure in E2 Node after each occurrence of a defined RIC Subscription procedure Event Trigger.

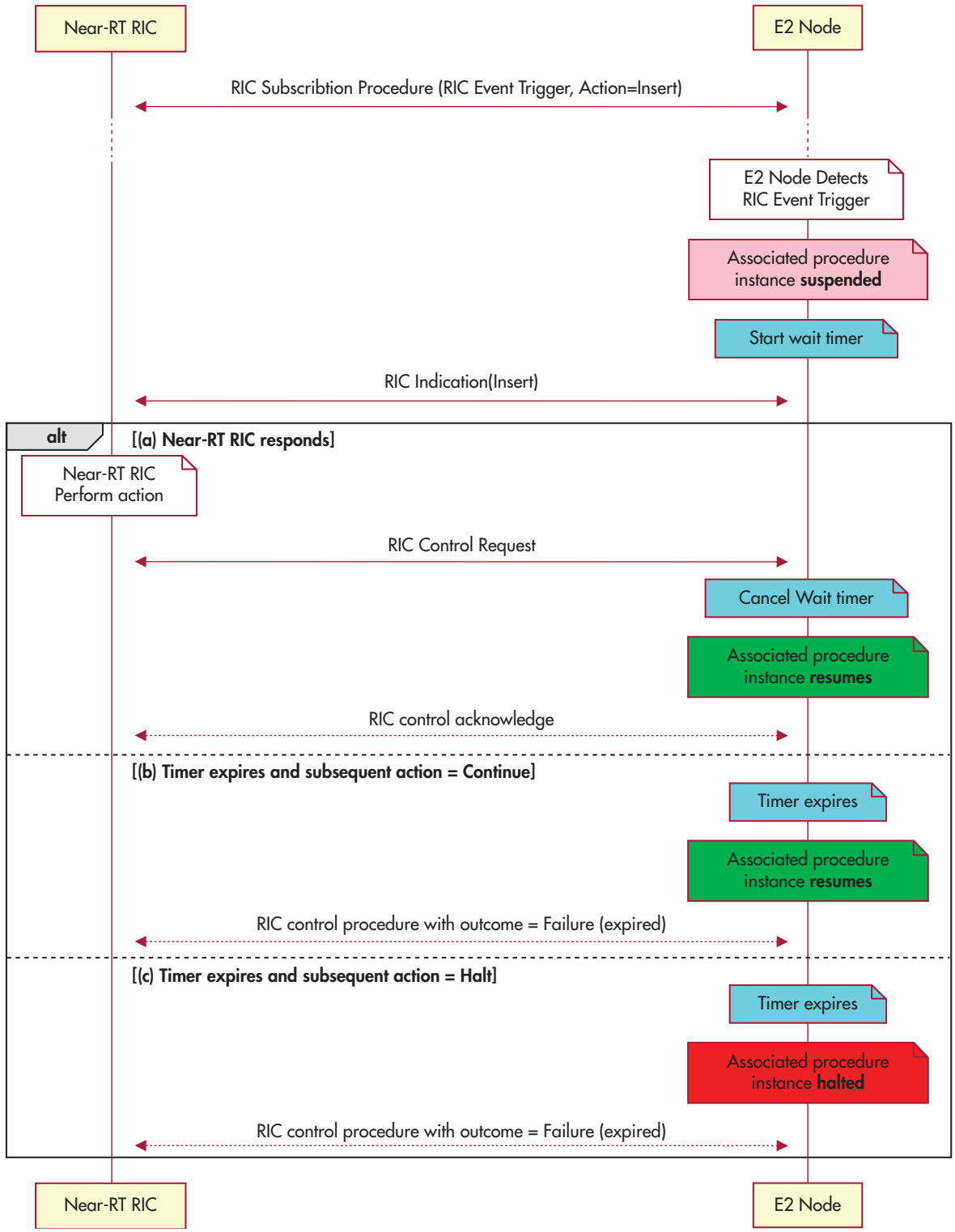


Figure 6: RIC Indication Procedure

2.7.3 Control Service:

Near-RT RIC sends a Control message to the E2 Node to initiate a new associated procedure or resume a previously suspended associated procedure in the E2 Node.

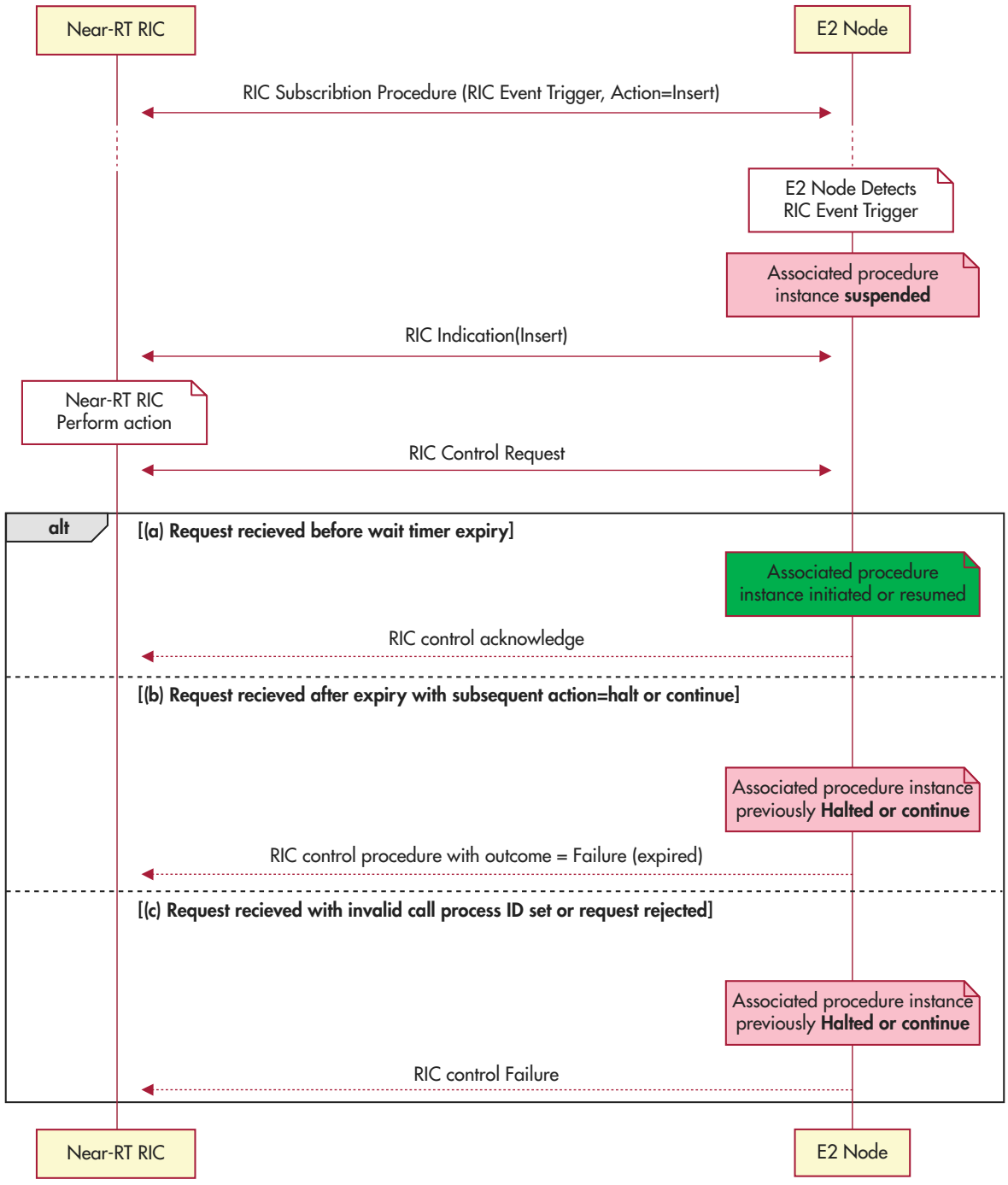


Figure 7: RIC Control Procedure

2.7.4 Policy Service:

Near-RT RIC requires that the E2 Node executes a specific POLICY during the functioning of the E2 Node after each occurrence of a defined RIC Subscription procedure Event Trigger.

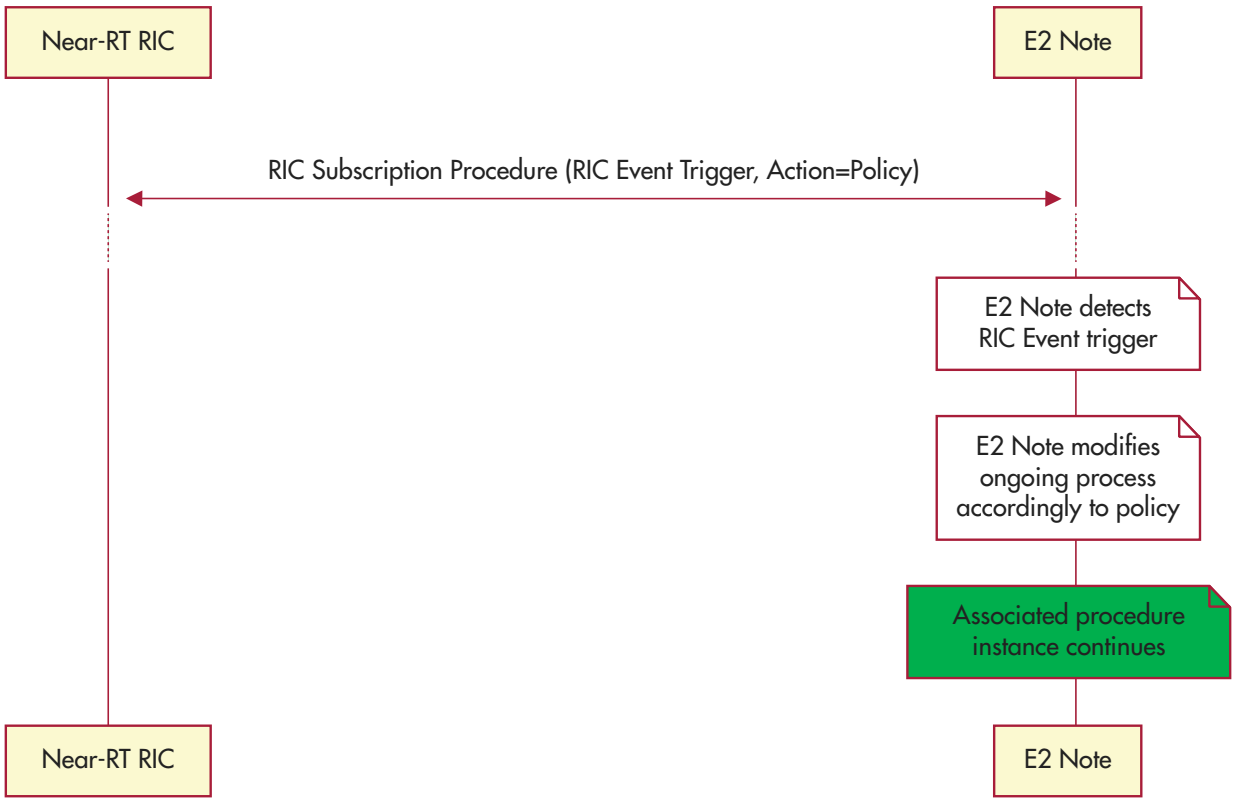


Figure 8: RIC Policy Procedure

2.8 Deployment Considerations

2.8.1 Deployment Topology

Deployment and operation of RAN in the 5G era is becoming complex due to various use cases and applications supported in a network as well as diverse combinations of network parameters and configurations. Furthermore, introduction of split architecture and virtualizations will increase the complexity of RAN. Under these circumstances, it has become increasingly difficult to manage RAN deployment and operations as well as to achieve RAN optimization with traditional manual operations. In order to solve this issue, introduction of intelligence in RAN is inevitable, enabling automated management and control by using big data analysis, artificial intelligence (AI) and machine learning (ML).

From the operators' perspective, one of the important benefits of introducing intelligence is to reduce OPEX through Digital Transformation in the RAN operation, which reduces associated operational activities and cost such as drive tests, manual configuration and optimization. Another important benefit is improved RAN performance through automated optimization of radio resource management and control, which will contribute to improvement of customer satisfaction and creation of new businesses. Below figure illustrates the placement of RIC in the network deployment.

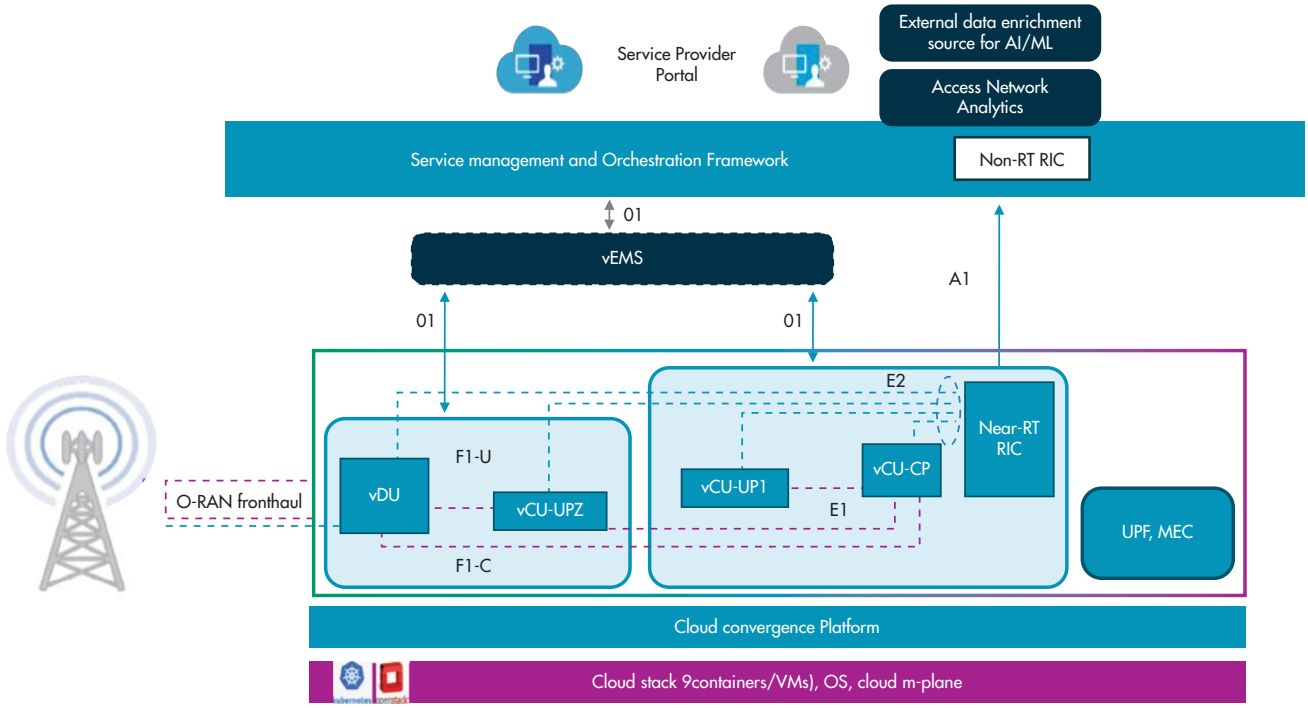


Figure 9: Deployment Topology for RIC

3 - Architecture

3.1 Near-RT RIC System Architecture

Overall Architecture is already covered as a part of introduction. Below details captures the internal architecture of near-RT RIC

The Near-RT RIC hosts the following functions:

- Database, which allows reading and writing of RAN/UE information;
- xApp subscription management, which merges subscriptions from different xApps and provides unified data distribution to xApps;
- Conflict mitigation, which resolves potentially overlapping or conflicting requests from multiple xApps;
- Messaging infrastructure, which enables message interaction amongst Near-RT RIC internal functions;
- Security, which provides the security scheme for the xApps;
- Management services;
    - » Fault management, configuration management and performance management as a service producer to SMO;
    - » Life-cycle management of xApp
- Logging, tracing and metrics collection, which capture, monitor and collect the status of Near-RT RIC internals and can be transferred to external system for further evaluation;
- Interface Termination
    - » E2 termination, which terminates the E2 interface from an E2 Node;
    - » A1 termination, which terminates the A1 interface from the non-RT RIC
- O1 termination, which terminates the O1 interface from SMO;
- Functions hosted by xApps, which allow services to be executed at the Near-RT RIC and the outcomes sent to the E2 Nodes via E2 interface.

This is summarized in the figure below.

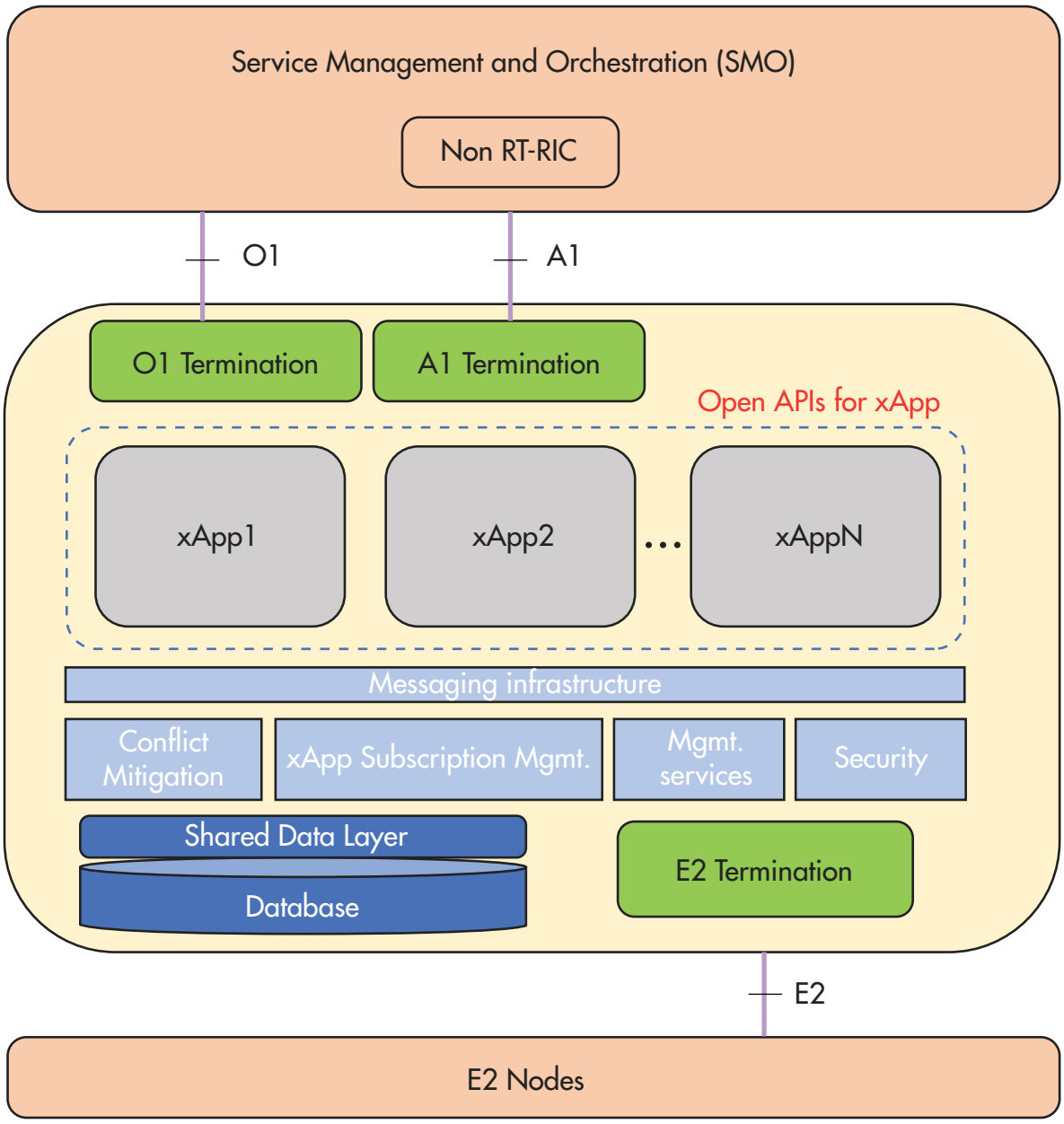


Figure 10: Near-RT RIC Internal Architecture

3.2 Non-RT RIC System Architecture

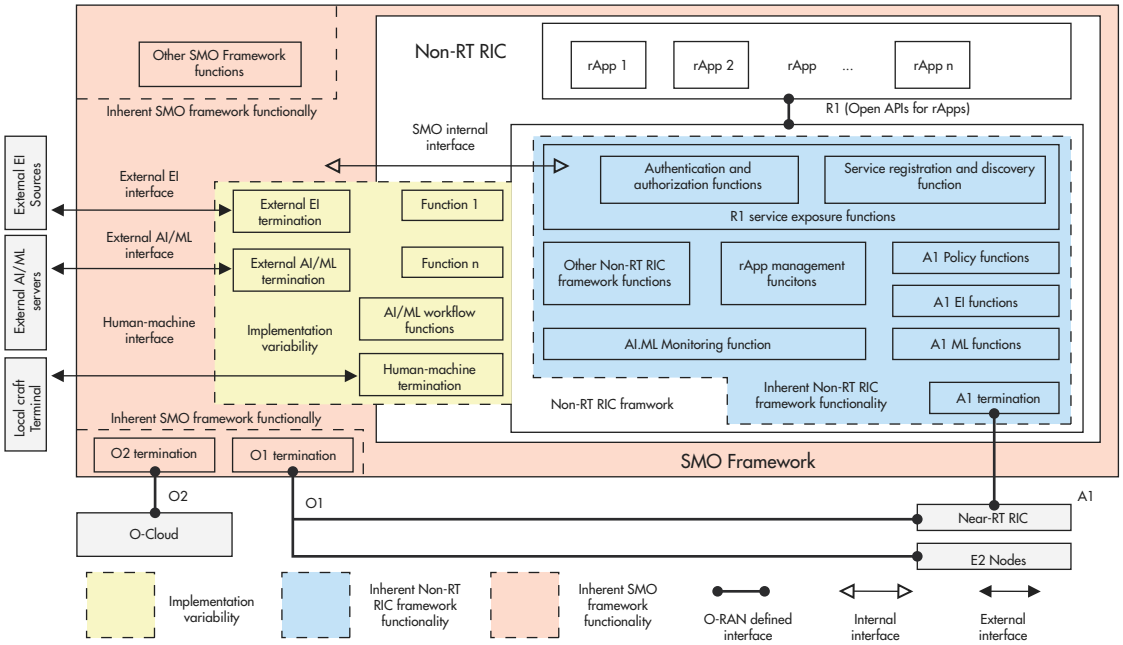


Figure 11: Non-RT RIC Architecture Functional View

Above figure illustrates the Non-RT RIC architecture. Non-RT RIC is an internal functionality of SMO framework and the diagram shows three categories of components of Non-RT RIC: rApps, Non-RT RIC framework and Open APIs for rApps.

- rApp is an application designed to run on the Non-RT RIC.
- Non-RT RIC framework is a collection of Non-RT RIC framework functions.
- R1 interface (Open APIs for rApps) are Non-RT RIC internal interface between rApps and Non-RT RIC framework and it is a collection of services, such as service registration and discovery services, AI/ML workflow services and A1-related services. Note that whether the function that provides the services is in Non-RT RIC framework or in SMO framework is transparent to the open APIs.

Note that one Non-RT RIC can connect to multiple Near-RT RICs.

A function is a logical entity that plays the roles of services producer and/or service consumer.
Inside the Non-RT RIC framework, there is a set of essential Non-RT RIC framework functions, which is illustrated as the blue area in the diagram. Because these functions are used to support A1 interface and rApps, it is natural to deploy these functions inside Non-RT RIC framework. Such functions are denoted as "inherent Non-RT RIC framework functionality".

To facilitate modular rApps, "rApp management functions" and "R1 service exposure functions" are required. "rApp management functions" includes rApp conflict mitigation. This example of rApp management is regarded as inherent to Non-RT RIC framework. Note that functions for rApp orchestration are a not part of rApp management functions and rApp orchestration functions can be a part of SMO framework.

Three terminations of external interfaces are demonstrated as examples of "implementation variable" functions: external EI termination, external AI/ML termination and human-machine termination. External EI termination is connected to external EI sources to import enrichment information for Non-RT RIC applications. External AI/ML termination is connected to external AI/ML server for ML model importation. Human-machine termination is used to inject RAN intent manually.

4 - Evaluation

4.1 Open-Source Evaluation for SDRAN – µONOS Near-RT RIC Solution

4.1.1 System Architecture

Similar to O-RAN, SD-RAN also supports both horizontal and vertical RAN disaggregation. Horizontal disaggregation effectively splits the RAN protocol stack so that the individual components can be realized independently. This aims to deal with the challenges of high total cost of ownership, high energy consumption, better system performance by intelligent and dynamic radio resource management, as well as rapid, open innovation in different components while ensuring multi-vendor operability. Vertical disaggregation focuses on control/user plane separation (CUPS) of the CU, further disaggregating it into CU-U, the logical node hosting the userplane part of the PDCP protocol and SDAP protocol and CU-C, the logical node hosting the control-plane part of the PDCP protocol and the RRC protocol.

The third-tier of disaggregation follows the SDN paradigm by carrying vertical disaggregation one step further. It does this by separating most of RAN control (RRM functions) from the disaggregated RAN components (mainly from CU-C), and logically centralizing them as applications running on an SDN Controller, which is labelled as the near Real-time RAN Intelligent Controller (nRT-RIC) in the O-RAN Architecture.

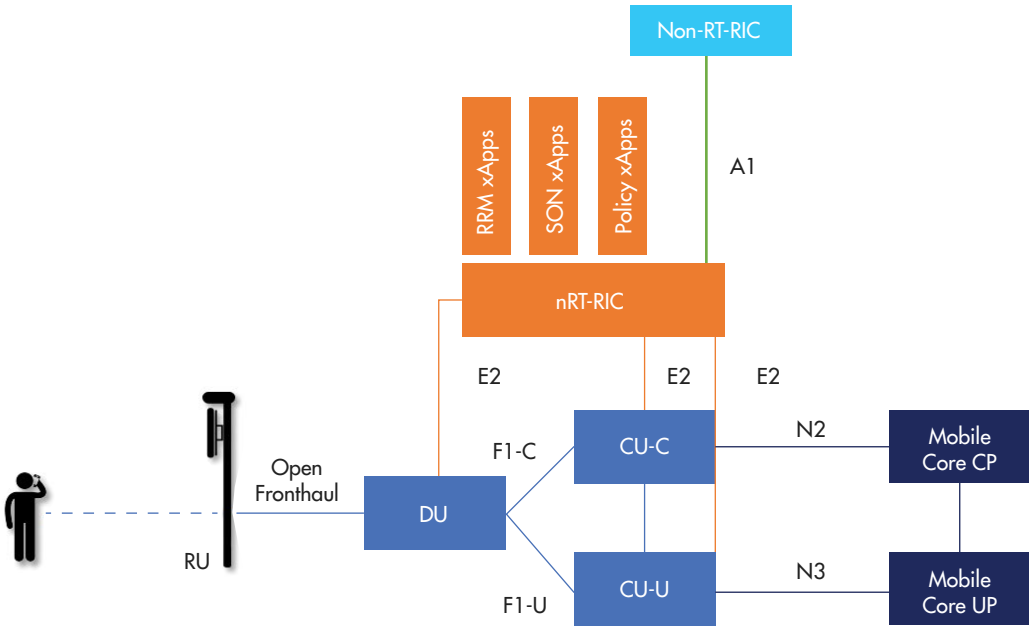


Figure 12: RAN Disaggregation – Overall Scenario

The SD-RAN Architecture is illustrated in the following figure:

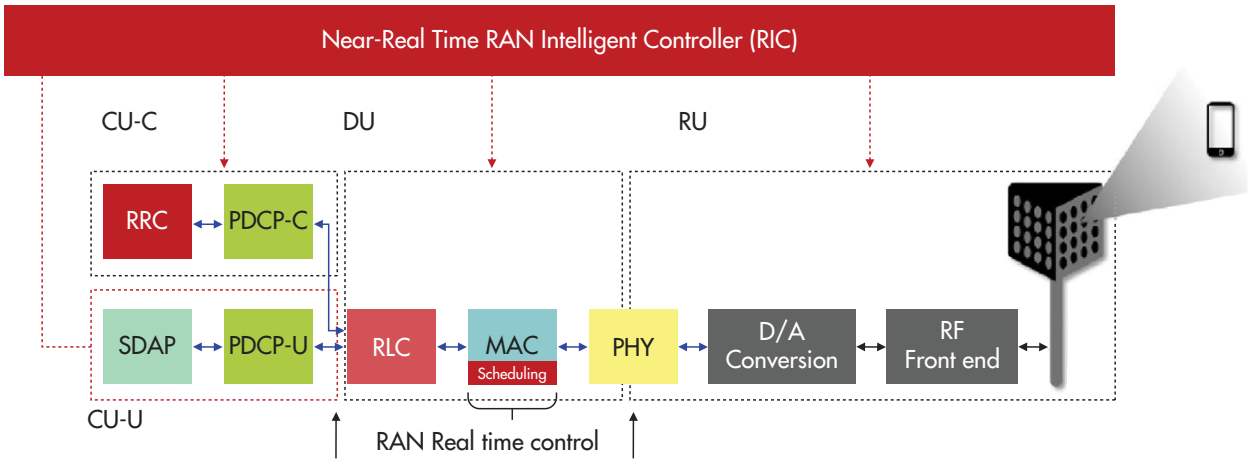


Figure 13: Software Defined RAN Control

It is to be noted here that the MAC-side RRM intelligence (RAN Real Time Control) is not centralized in SD-RAN architecture, but rather, it continues to run in a distributed manner across the geography. However, using an open interface, SD-RAN allows for these functions to be configured in real-time by the Near-Real-Time RIC. SD-RAN solution basically talks about near-RT RIC that uses ONOS-based SDN controller. It is based on a microservices architecture (ONOS RIC) supporting both ONF xAPPs as well as 3rd part xAPPs. It supports a new generation of control and configuration interfaces and standards: P4/P4Runtime, gNMI/OpenConfig, gNOI.

It is cloud native, adopting best practices in micro- serviceswith the use of polyglot interface mechanism (gRPC) and container management (Kubernetes). The complete solution diagram is illustrated in the Figure 7. The High-level ONOS RIC Architecture is explained in Figure 8.

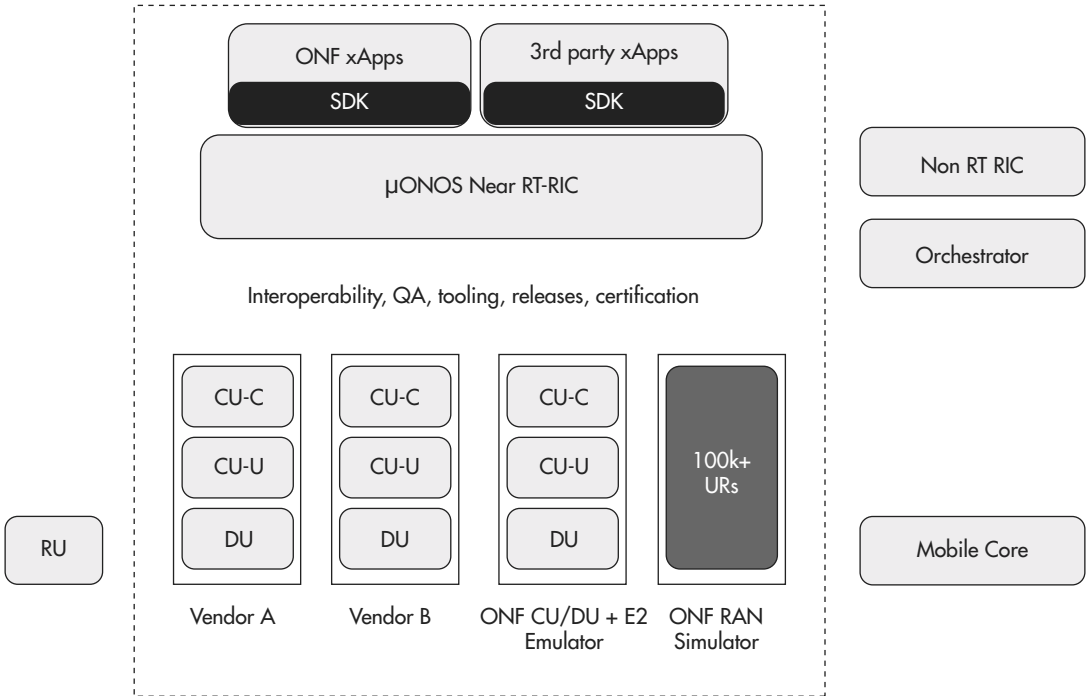


Figure 14: SD-RAN Solution

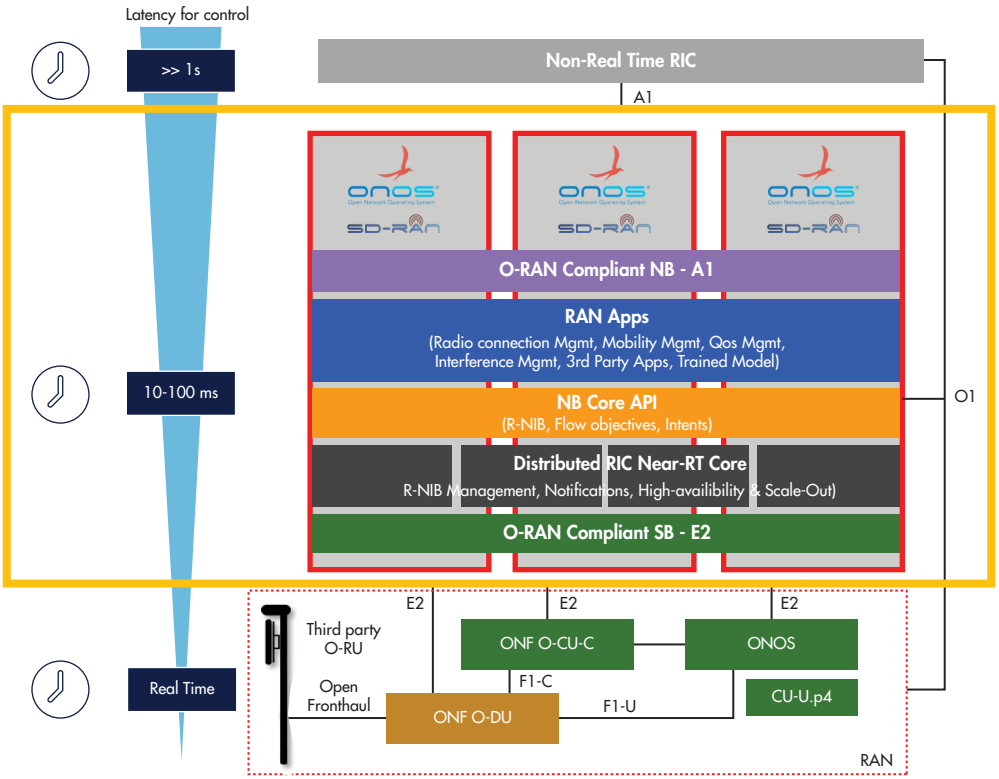


Figure 15: High-level ONOS RIC Architecture

ONOS RIC enables a multi-cluster operation for high-availability and scalability. The southbound interface of ONOS RIC is the O-RAN specified E2 interface. ONOS provides a distributed data store to maintain the Radio Network Information Base (R-NIB), messaging infrastructure as well as topology, control and configuration managers in a microservices environment. It intends to support conflict resolution to resolve conflicts emanating from multiple RAN applications. ONOS RIC provides an open API to host 3rd party RAN applications. These applications will vary from basic RRM functions, to Self-Organizing Network (SON) applications, to ML driven network optimization applications.

Following figure illustrates the SD-RAN micro-ONOS near-RT RIC components

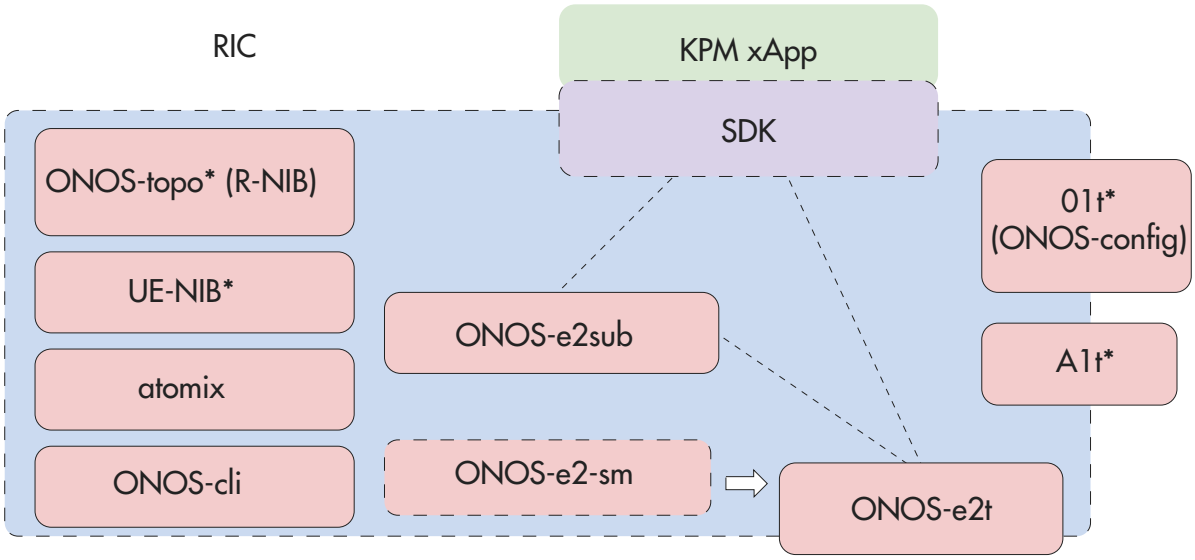


Figure 16: SD-RAN near-RT RIC Components

4.1.2 Configuration Overview

The ONF community comes up with both OAI based SD-RAN/Test framework as well as RANSim based test framework. Appropriate configurations to be defined (as per the support provided by OAI and RANSim). While OAI based test setup can be used to ensure practical deployment and testing of real software/ hardware, RANSim can be used to study the scalability, use case development, application testing, AI/ML based solutions and future research etc.

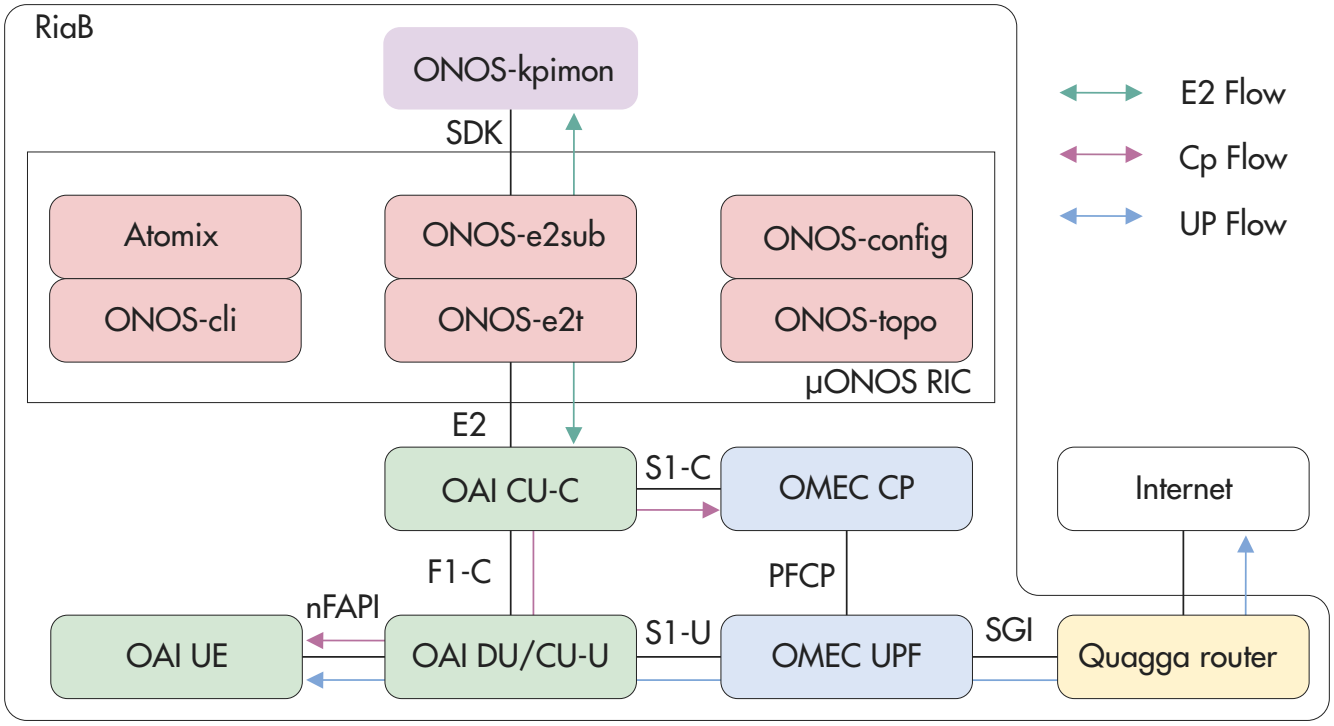


Figure 17: OAI based SD-RAN Dev/Test Setup

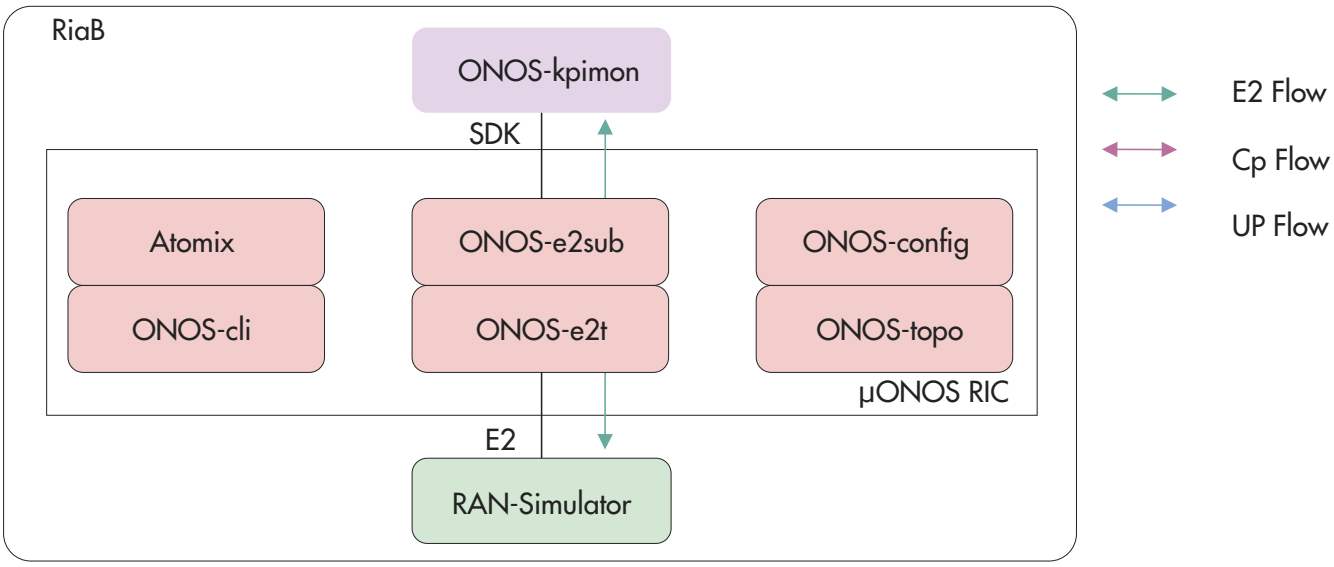


Figure 18: RANSim based SD-RAN Dev/Test Setup

4.1.3 Feature Set Supported

1. ORAN Interface E2 Termination

The µONOS RIC supports E2 Application Protocol over ASN.1/SCTP to all the RAN Elements on the southbound and internally converts into ProtoBuff over gRPC protocol.

The E2 termination microservice, which is part of the RIC, talks to the RAN components and is capable of consuming O-RAN defined Service Models.

2. SDK for supporting xApps

SDK, currently developed in Go-Lang/Python supports both,

- In-house xApps, which are developed by ONF itself
- 3rd party xApps

The SDK helps in standardizing the code and in portability of xApp's across RIC platforms, such as:

- ONF RIC
- OSC (O-RAN software community) RIC

The Go-Lang based SDK is developed by ONF team itself. The Python SDK is contributed by Facebook. This can be very useful to run AI/ML applications as there are many Python based ML libraries which can be used right away. Goal of the SDK is to make it simple for the development and testing of the Apps. The other important objective is to make the xApps portable across the RIC platform, as described above.
SDK usually has:

- **The RIC-Platform independent layer:** This encodes and decodes the standard service models and provides extended functionality which need not be specific to the RIC.
- **The RIC-Platform specific layer/Portability Layer:** This layer terminates/abstracts the protocols implemented by RIC-specific vendor, which is mainly gRPC protocol in the case of ONF RIC and would be ASN.1 over TCP for OSC RIC.

3. O-RAN Service Models

O-RAN defines Service Models which contains information related to various use-cases. A SM typically contains triggers and actions and serves as APIs exposed by CU/DU/RU. It's a contract between RIC and RAN elements and defines how they interface with each other.
Few examples are:

- SDRAN v1.0 supports KPM 1.0 SM and
- SDRAN v1.1 supports KPM 2.0.3 SM

Some of these Service Models which are not backward compatible do lead to integration concerns.

4. Reference White Box Solution for Radio Elements

Using Intel Hardware and a Software define Radio Box (USRP) from National Instruments and with antennae, this hardware can talk to Phones and UE Software which runs on a different White box. This solution comes from Open Air Interface project-based LTE CU/DU. It's enhanced by ONF with the E2 Interface, by which it can support the O-RAN standard to the RIC. This could potentially be integrated with any other O-RAN based RIC's as well.

5. RAN Simulator

The RAN Simulator helps in providing the characteristics of RAN, as defined by the O-RAN interface specification between the RAN and the RIC. This currently means the E2 interface exposed by the CU and eventually could be the DU interface as well. This could potentially be integrated with any other ORAN based RIC's as well.

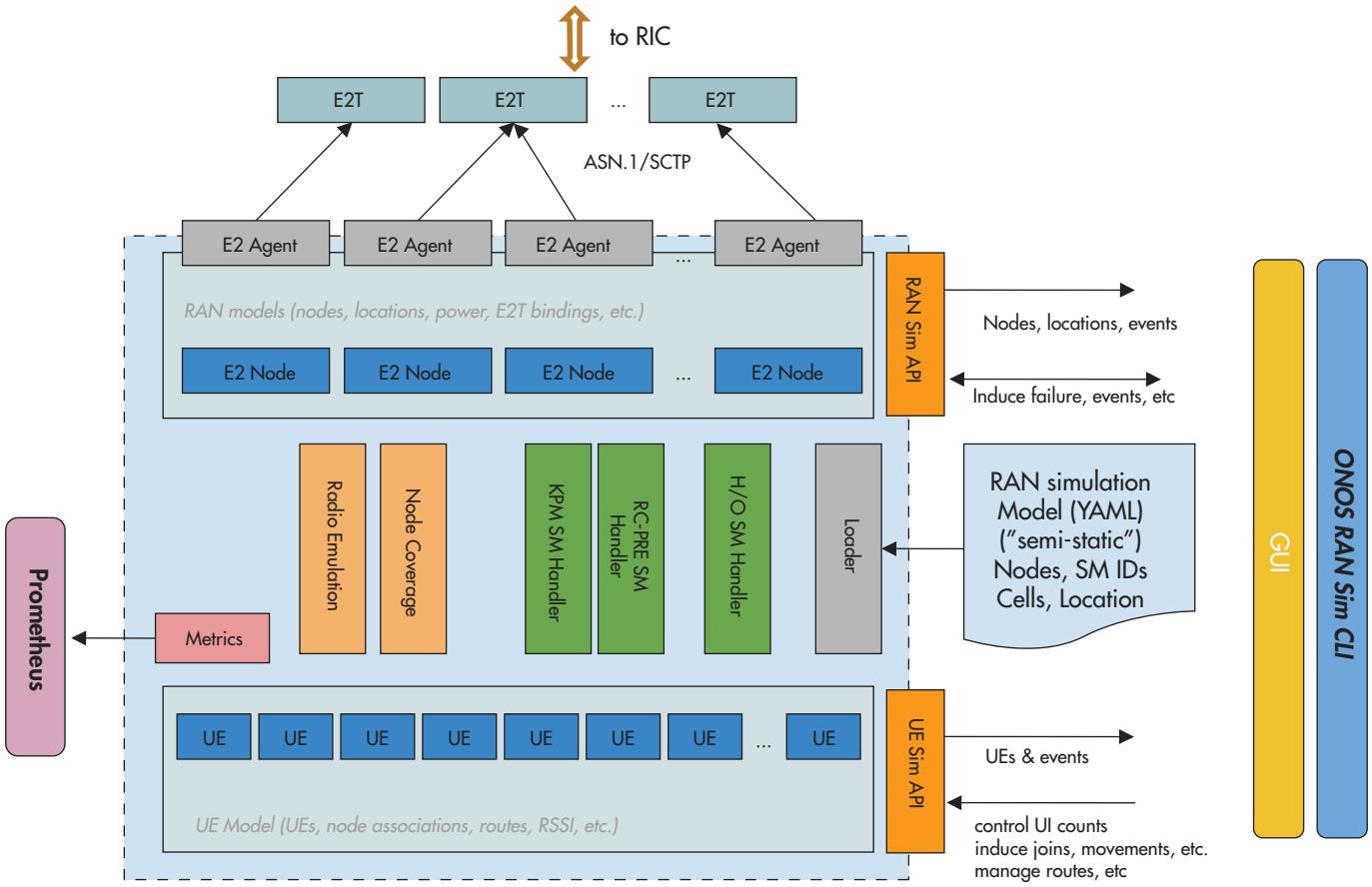


Figure 19: RAN Simulator Architecture

4.1.4 Hardware requirements by SD-RAN

ONF RIC is a containerized, fully virtual software. All of it is managed and installed using Helm charts, which is the package manager for Kubernetes.

In addition to the RIC, ONF also provides a white-box RU/DU/CU component, whose software comes from OAI Project, which has been enhanced to add the O-RAN specific interface.

This is specifically, the E2 agent to talk to RIC and runs on CU-C. This in turn integrates with the ONF developed mobile core – OMEC.

The "SD RAN in a box" option - Avoids specific hardware requirement. It can be run on any commodity hardware. That means RAN side can run as a simulator on a VM. Intel NUC is used to run OAI UE software to Software defined Radio box (USRP) which can simulate a UE. USRP box can also be used to run RU. Intel NUC is used to run the OAI DU-CU combo module too.

Near- Real Time RIC would be hosted on Edge Cloud.

ONF's SD-RAN is an open exemplar platform that is cloud-native and is built on ONF's well-established, operator- approved and deployed platforms, such as:

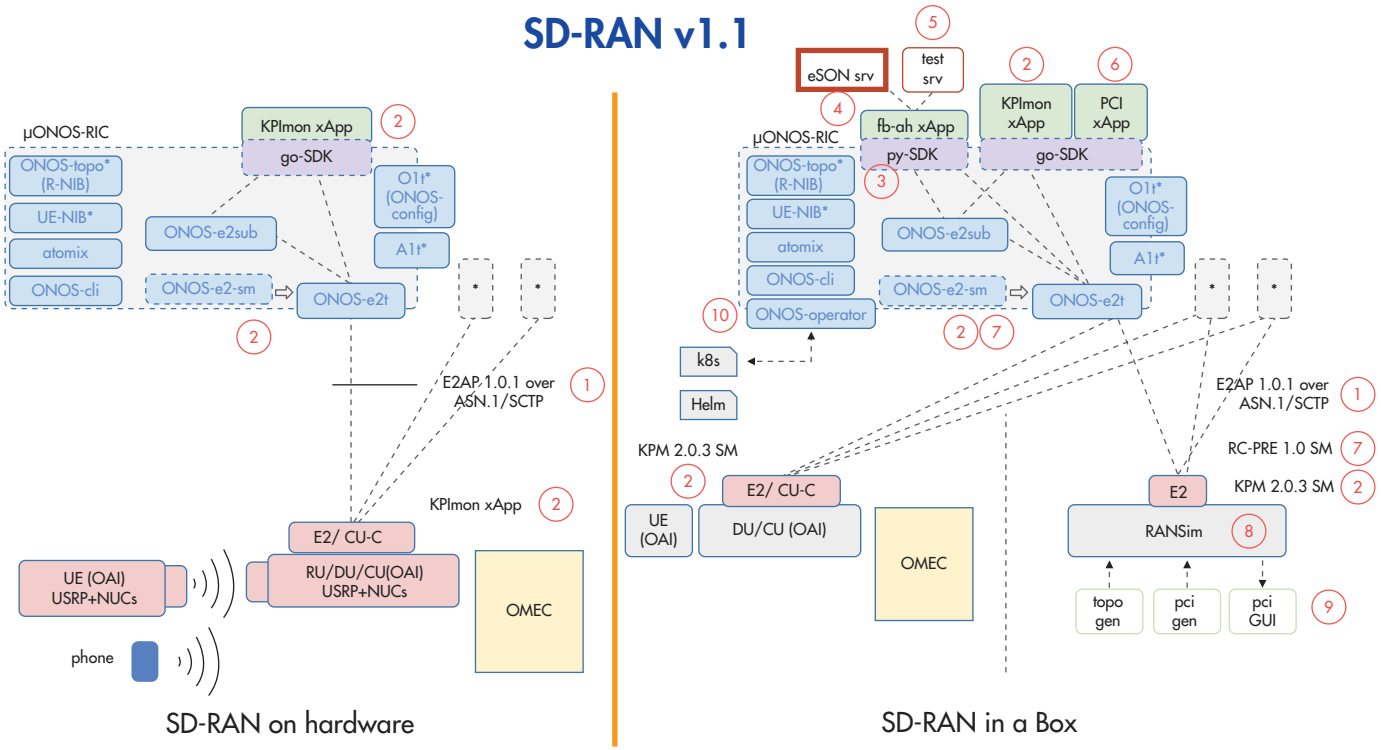


Figure 20: SD-RAN Testing Setup (v 1.1, Released in May 2021)

4.1.5 Current Limitations

The µONOS RIC is started recently, around August 2020 and have had few releases. Currently it supports only Near-RT RIC.

We need to wait for the feedback from the Operators about the flexibility and advantages it delivers by integrating home grown xApps and vendors. Though the group is currently members-only group for contributing and checking the code, it is likely to be Open-Sourced to non-members in 2021. The languages which the SDK supports are Golang and Python.

4.1.6 Gap Analysis

The flexibility of ONOS controller, integration of 3rd part xApps (non-member developed xApps) are still the key points which the ONF community has to demonstrate. Further the practical deployment of the exemplar platform for vendor agnostic solution design is to be evaluated critically.

SD-RAN currently supports xApps in Golang and Python. If more Programming languages are added, it might help in the proliferation of the xApps.

4.1.7 Roadmap

Three Specific near-term deliverables are planned by ONF Community related to SD-RAN development and integration:

- Contributing to O-RAN alliance
- Collaborating with O-RAN SC
- Seeding platform with TIP

ONF also plans to include the following points while moving from 4G to 5G PCI:

- Mobility management, load balancing and handover
- New use cases
- SD-RAN will prototype extensions to the data models for the E2 & A1 interfaces to enable flexibility & support for a broad set of xApp functions and applications, with a specific goal of enabling a robust spectrum of RRM & SON functionality within xApps.
- The SD-RAN project plans to contribute an SDK for the northbound interfaces from the nRT-RIC towards the xApps to support & promote availability of interoperable xApps and rApps.
- SD-RAN exemplar xApps will help demonstrate xApp interoperability between OSC NR-RIC & ONF µONOS-RIC.

4.1.8 System Integration

From the version v1.1 onwards, there has been lot of interest to contribute onto the list of xApps, by 3rd party vendors such as AirHop. The commercial Application called eSON system by AirHop easily was integrated with the help of xApp adaptor, developed by Facebook.

This xApp used the Python based SDK and integrated it into the eSON system with very minimal code changes done on eSON, using the AirHop defined gRPC based interface. The eSON is a cloud-based service and can interact with the entire ONOS RIC, which is virtualized. ONF also introduces SD-Core, an Open-Source project for building a 5G/4G disaggregated mobile core optimized for emerging private 5G and enterprise use cases. This is complemented by SD-Fabric, an Open-Source, P4-programmable hybrid cloud network fabric with the power to push customized packet processing deep into networking elements. Think of SD-Fabric as a cloud-managed, full-stack, programmable network fabric for edge applications for Industry 4.0 powered by 5G.

4.2 Open-Source Evaluation for ORAN-SC RIC Solution

4.2.1 System Architecture

4.2.1.1 Near-RT RIC Platform

The Near-RT RIC Platform is a software based Near-real time micro-service-based platform for hosting micro-service-based applications - the xApps - that run on the near-RT RIC platform.

xApps are not part of the RIC platform and developed in projects that are separate from the near-RT RIC platform project. The near-RT RIC platform is providing xApps the infrastructure for controlling a distributed collection of RAN base stations (eNB, gNB, CU, DU) in a region via the O-RAN alliance's E2 protocol ("southbound"). As part of this infrastructure, it also provides "northbound" interfaces for operators: the A1 and O1 interfaces. Using the O-RAN alliance's A1 interface operators can express their intent for the network behaviour and collect feedback on the implementation status of this intent. The RIC platform also is to implement the O-RAN alliance's O1 management interface primarily integrating FCAPS capabilities with ONAP. The near-RT-RIC platform provides mediation for both the E2, A1, and O1 interfaces between the xApps and the RAN elements (E2) and between the xApps and the operator (A1, O1), respectively. xApps use the services of the RIC platform.

Micro-services that are needed for the near-RT RIC platform shall be considered part of the near-RT RIC Platform, including lifecycle management of xApps, restricting the span of control of xApps, xApps composition, xApps conflict resolution, E2 status information and basic RAN data as obtained over the base E2 protocol (but not via E2 functions specific to the E2 services implemented on top of the E2 base protocol), logging and software execution tracing, configuration management, in-memory database add-ons, messaging services, statistics collection, micro-service scheduling and security.

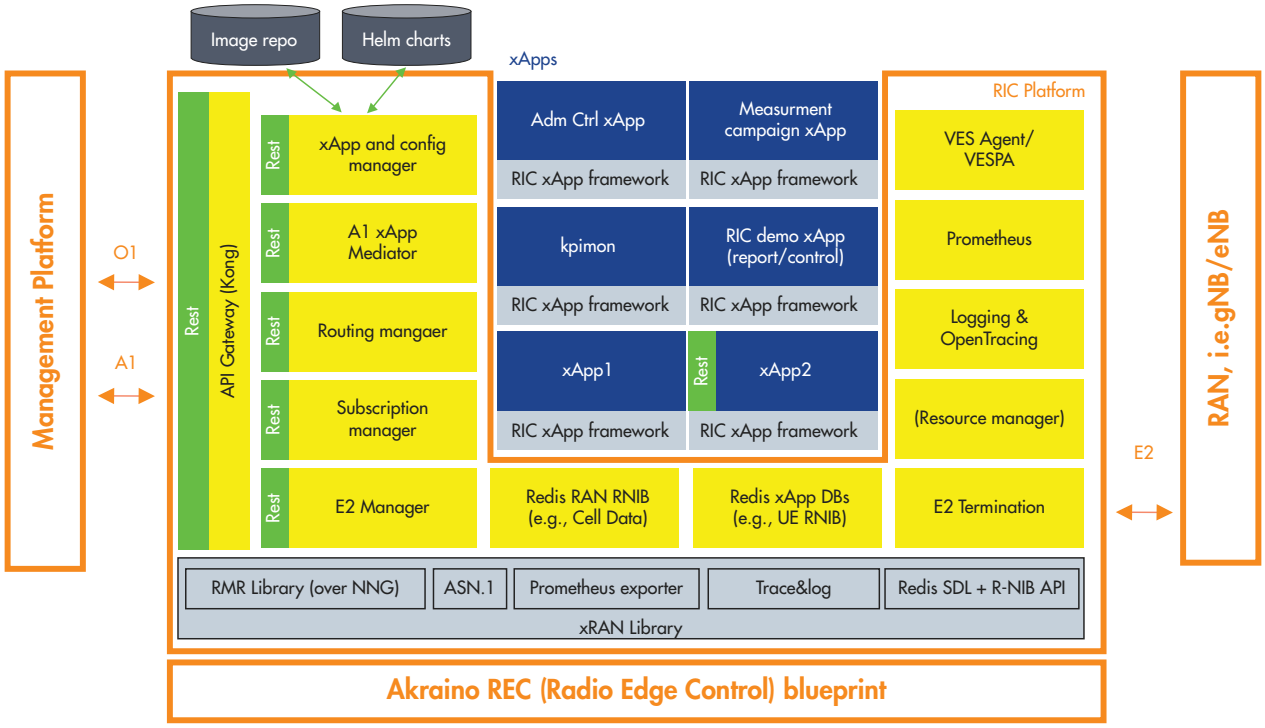


Figure 21: ORAN-SC Near-RT RIC Components

The RICP (near-RT RIC platform) includes the following component repositories:

O1 mediator:

- Supports NETCONF
- xApp health status
- Simple xApp configuration via pass-through YANG models.
- E2 states of connected gNBs
- Alarm states (incl. xApps) from new alarm adapter
- O1 Alarm events from new alarm adapter to Prometheus Alert Manager to VESPA via REST to external VES collector

A1 mediator

- A1 mediator uses SDL+Redis for persistence
- A1 mediator supports Prometheus for interface statistics

RMR (RIC message routing)

- more reliable route distribution with routing manager
- rewrite replacing nanomsg-next-gen with a low-latency RIC-specific implementation

Redis and SDL (Shared Data layer):

- Deployed in HA deployment by default (using Sentinel).

E2 termination

- Supports Prometheus for interface statistics

E2 manager

- Support big red button: close all E2 connections, no new E2 connections
- Manages mapping of E2 termination instance to E2 node and provides the information to the routing manager and RMR.

xApp framework APIs

- Compatible with C++, Go and Python

4.2.1.2 xApp Architecture:

The architecture of an xApp consists of the code implementing the xApp's logic and the RIC libraries that allow the xApp to:

- Send and receive messages.
- Read from, write to and get notifications from the SDL layer.
- Write log messages.

Furthermore, xApps can use access libraries to access specific name-spaces in the SDL layer. For example, the R-NIB that provides information about which E2 nodes (e.g., CU/DU) the RIC is connected to and which SMs are supported by each E2 node, can be read by using the R-NIB access library.

The O-RAN standard interfaces (O1, A1, and E2) are exposed to the xApps as follows:

- xApp will receive its configuration via K8s ConfigMap - the configuration can be updated while the xApp is running and the xApp can be notified of this modification by using inotify ()
- xApp can send statistics (PM) either by (a) sending it directly to VES collector in VES format, (b) by exposing statistics via a REST interface for Prometheus to collect
- xApp will receive A1 policy guidance via an RMR message of a specific kind (policy instance creation and deletion operations)
- xApp can subscribe to E2 events by constructing the E2 subscription ASN.1 payload and sending it as a message (RMR), xApp will receive E2 messages (e.g., E2 INDICATION) as RMR messages with the ASN.1 payload. Similarly, xApp can issue E2 control messages.

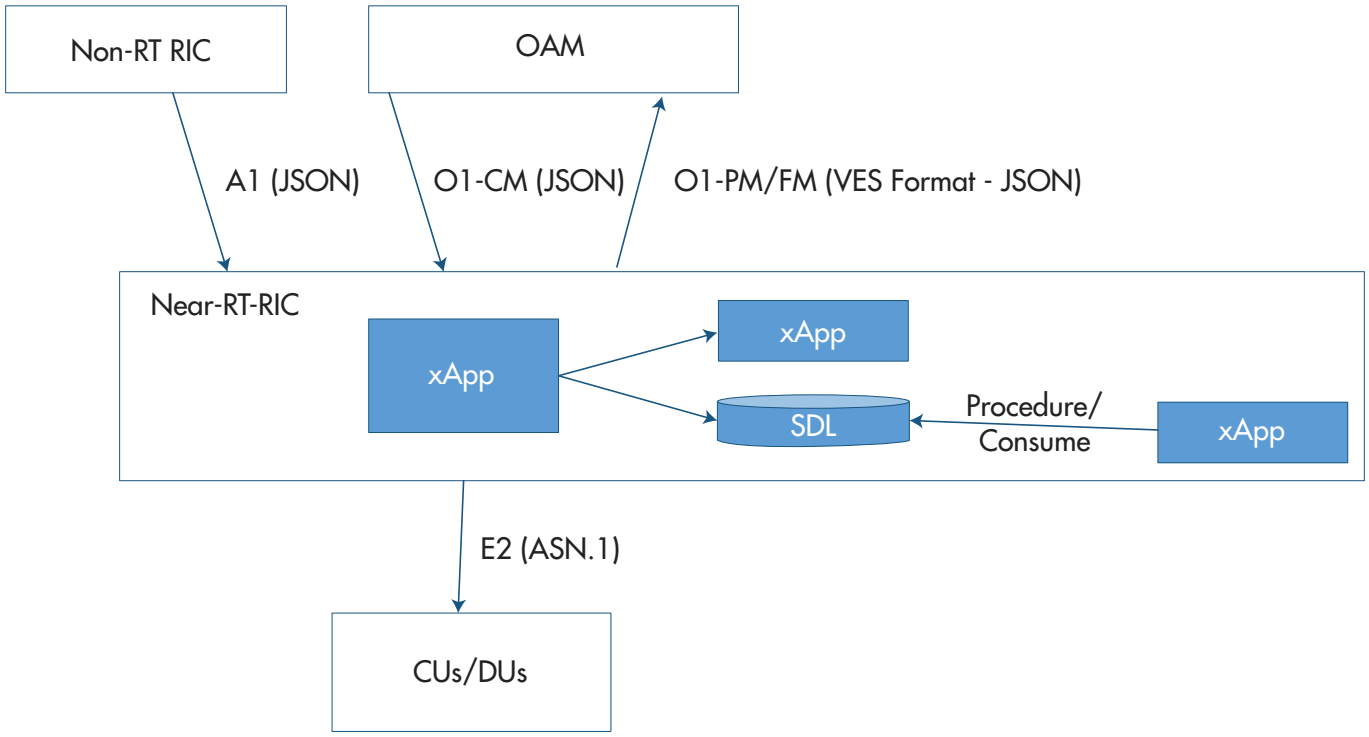


Figure 22: xApp in the context of O-RAN Architecture

In addition to A1 and E2 related messages, xApps can send messages that are processed by other xApps and can receive messages produced by other xApps. Communication inside the RIC is policy driven, that is, an xApp cannot specify the target of a message. It simply sends a message of a specific type and the routing policies specified for the RIC instance will determine to which destinations this message will be delivered (logical pub/sub).

4.2.1.3 Non-RT RIC Platform

The Non-Real Time RIC (RAN Intelligent Controller) is an Orchestration and Automation function described by the O-RAN Alliance for non-real time intelligent management of RAN (Radio Access Network) functions. The primary goal of the Non-RT RIC is to support non-real time radio resource management, higher layer procedure optimization, policy optimization in RAN and providing guidance, parameters, policies and AI/ML models to support the operation of near-Real Time RIC functions in the RAN to achieve higher-level non-real time objectives. Non-RT RIC functions include service and policy management, RAN analytics and model-training for the near-Real Time RICs. The non-Real Time RIC project provides concepts, specifications, architecture and reference implementations as defined and described by the O- RAN Alliance architecture.

The Non-RT RIC implementation will communicate with near-Real Time RIC elements in the RAN via the A1 interface. Using the A1 interface, the Non-RT RIC will facilitate the provision of policies for individual UEs or groups of UEs; monitor and provide basic feedback on policy state from near-Real Time RICs; provide enrichment information as required by near-Real Time RICs; and facilitate ML model training, distribution and inference in cooperation with the near-Real Time RICs.

As shown in the Component Architecture diagram below, the Non-RT RIC functions partly leverage and extend some existing infrastructure from ONAP to support non-Real Time control of the RAN (Radio Access Network).

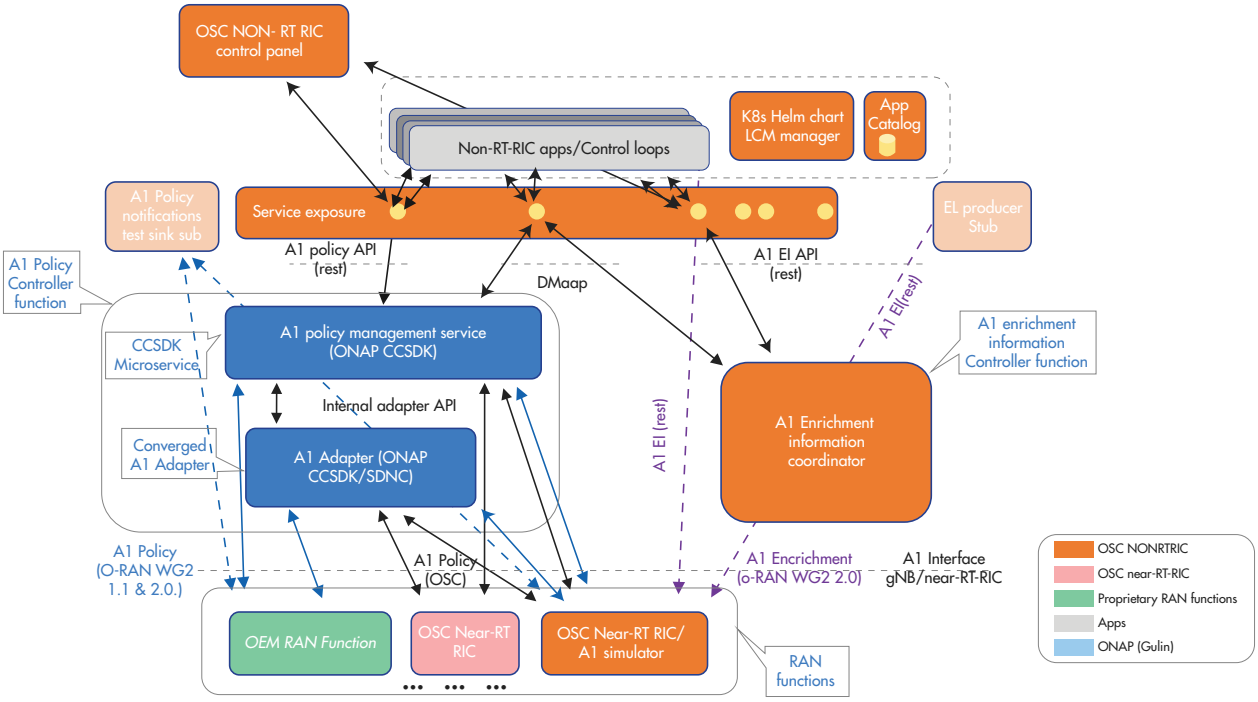


Figure 23: Functional view of OSC Non-RT RIC

Non-RT RIC components:

1. Non-RT-RIC Control Panel:

Graphical user interface to interact with the NON-RT RIC services.

- View and Manage A1 policies in the RAN (Near-RT-RICs)
- Graphical A1 policy creation/editing is model-driven, based on policy type's JSON schema
- View and manage producers and jobs for the Enrichment coordinator service
- Configure A1 Policy Management Service (add/remove Near-RT-RICs)
- Interacts with the A1-Policy Management Service & A1-EI-Coordinator (REST NBIs) via Service Exposure gateway

2. Non-RT-RIC (Spring Cloud) Service Gateway:

Spring cloud Gateway provides the library to build the API Gateway for Micro-service architecture. In Non-RT-RIC we build the basic API gateway using spring cloud gateway which then exposes two Non-RT RIC functions; Policy Management Service & Enrichment Coordinator Service.

3. Non-RT-RIC (Kong) Service Exposure Prototyping:

Support Apps to use Non-RT RIC, SMO and other App interfaces

4. A1 Policy Management Service:

A1 Controller Service above A1 Adaptor that provides:

- Unified REST & DMaaP APIs for managing A1 Policies in all Near-RT-RICs
- Operations:
- Query A1 Policy Types in Near-RT-RICs
- Create/Query/Update/Delete A1 Policy Instances in Near-RT-RICs
- Query Status for A1 Policy Instances
- Maintains (persistent) cache of RAN's A1 Policy information
- Support RAN-wide view of A1 Policy information
- Streamline A1 traffic
- Enable (optional) re-synchronization after inconsistencies / Near-RT-RIC restarts
- Added support for multiple Near-RT-RICs (& multi-version support)
- Unified REST & DMaaP NBI
- Converged ONAP & O-RAN-SC A1 Adapter/Controller functions in ONAP SDNC/CCSDK (Optionally deploy without A1 Adaptor to connect direct to Near-RT- RICs)

5. Enrichment Information Coordinator

Coordinate/Register A1-EI Types, Producers, Consumers and Jobs.

- Maintains a registry of:
- A1-EI Data Types/schemes
- A1-EI Producers
- A1-EI Consumers
- A1-EI Jobs
- A1-EI Query API (e.g., per producer, per consumer, per types)
- Query status of A1-EI jobs
- Monitors all Near-RT-RICs and recovers from inconsistencies

6. Being extended to coordinate non-A1 Enrichment Information exchange between Non-RT-RIC AppsInitial Non-RT-RIC App Catalogue

Register for Non-RT-RIC Apps

- Apps can be registered/queried
- Limited functionality/integration for now

7. SDNC A1 Controller

Mediation point for A1 interface termination in SMO/Non-RT-RIC

- Implemented as CCSDK OSGI Feature/Bundles
- A1 REST southbound
- RESTCONF Northbound
- NETCONF YANG > RESTCONF adapter
- Mapping logic/Provider
- Can be included in any controller based on ONAP CCSDK

8. Near-RT RIC A1 Simulator

Stateful A1 test stub

- Used to create multiple stateful A1 providers (simulated Near-RT RIC)
- Supports A1-Policy and A1-Enrichment Information
- Implemented as a Python application
- Swagger-based northbound interface, so easy to change the A1 profile exposed (e.g., A1 version, A1 Policy Types, A1-E1 consumers, etc)

9. Initial K8S Helm Chart LCM Manager

Onboard, start, stop and modify Non-RT-RIC App µServices as Helm Charts

10. Test Framework

Function test is conducted to verify the functionality of the Non-RT RIC as well as the individual components. A test engine and a number of test script is used as a foundation for the test environment. The test environment uses docker images both the Non-RT RIC components as well as for the needed simulators

4.2.2 Configuration Overview

Below figure illustrates the end-to-end setup of ORAN-SC with all the RAN components from ORAN.

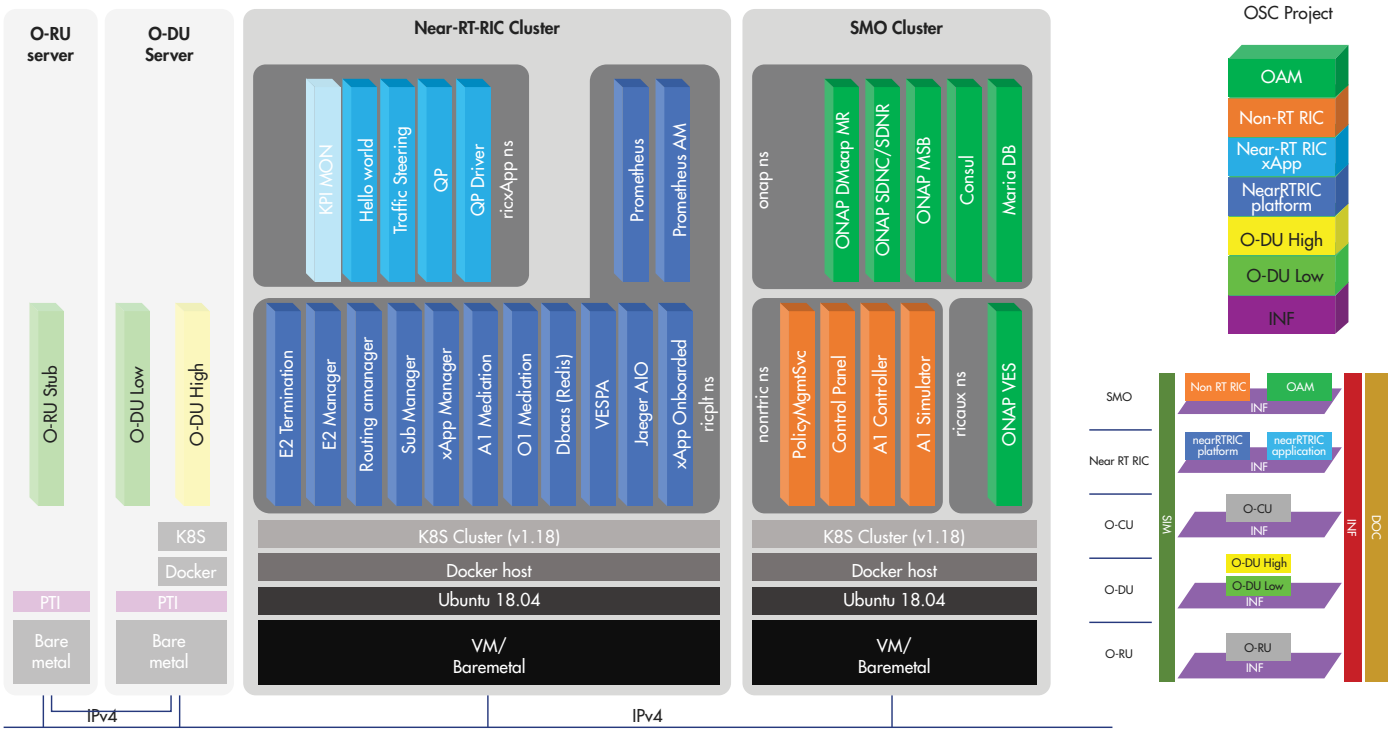


Figure 24: PoC Setup with RAN Components

RIC test and automation leverages Robot framework for testing and validating the RIC components Health check and E2E Performance testing of the RIC.

This test automation contains containers, Helm charts, Libraries and Test suites to support automated and end-to-end testing of the RIC. Below figure illustrates the test setup:

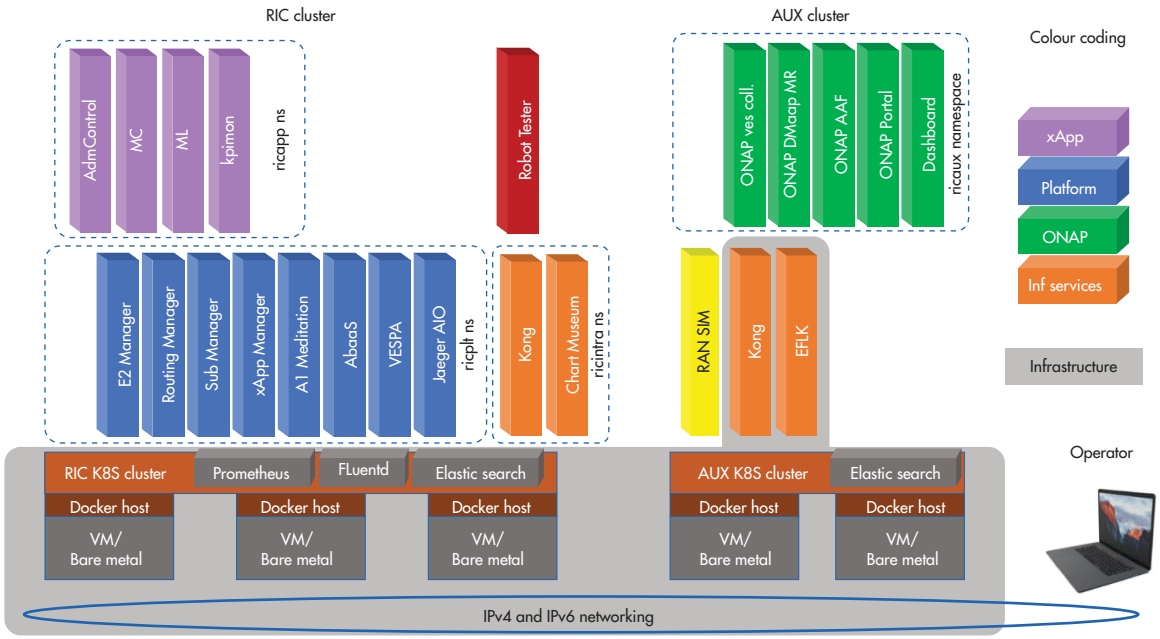


Figure 25: PoC Setup with Robot Tester/RAN SIM

4.2.3 Feature Set Supported

- The Near-RT RIC platform has implemented E2AP protocol as defined by the O-RAN alliance.
- The Bear-RT RIC O1 mediator implements the O1 interfaces based on NETCONF and YANG models, e.g., to check the health status of xApps.
- Enhanced RIC's messaging solution (RMR) to achieve higher throughput, various health check capabilities including a capability to ping E2 nodes via E2 and to do a health check of the A1 interface, better failure handling in various components.
- Operator-initiated E2 connection health check.
- Support of various xApp framework and SDL features in Python, C++ and Go.
- Support of basic traffic steering, quality prediction, anomaly detection and measurement campaign xApps.

4.2.4 Hardware requirements by ORAN-SC

4.2.4.1 Near-RT RIC:

The deployment of Near Real time RIC can be done on a wide range of hosts, including bare metal servers, OpenStack VMs and VirtualBox VMs. This section provides detailed instructions for setting up Oracle VirtualBox VMs to be used as installation hosts.

- OS: Ubuntu 18.04 LTS (Bionic Beaver)
- CPU(s): 4
- RAM: 16 GB
- Storage: 160 GB

4.2.4.2 AUX cluster:

To run the RIC-AUX cluster in a dev testing setting, the minimum requirement for resources is as below:

- OS: Ubuntu 18.04 LTS (Bionic Beaver)
- VM with 4 vCPUs
- 16G RAM
- 40G of disk space

4.2.4.3 Non-RT RIC and SMO:

- OS: Ubuntu 18.04 LTS (Bionic Beaver)
- CPU(s): 8
- RAM: 32 GB
- Storage: 160 GB

4.2.5 Current Limitations

There is no support to test the RIC in RAN in a box (similar to SD-RAN) environment. Open Test Framework and platform which provides standardized and self-serve service testing capabilities to test RIC components is still not made Open-Source. Most of the xApps from ORAN-SC are still at the basic stage and need some more time to mature.

4.2.6 Gap Analysis

A plan to make ORAN-SC based Near-RT RIC a minimum viable product is yet to be made. Test tools to harden the Near-RT RIC are currently at initial stage. No clear plans to make RIC platform a commercial deployment grade solution. xApp portability to different SDK is one grey area, where there is less traction.

4.2.7 Roadmap

- Implementation of Anomaly Detection (AD) xApp where the ML-model is trained using data from E2 simulator and then at runtime, using the data based, which can raise anomaly alarms.
- Quality Prediction (QP) xApp, true ML-based throughput prediction trained using data from simulator. QP receives prediction request for a set of UEs, determines the current UE metrics and current neighbour cells by reading the inFlux DB and provides prediction of the throughput prediction in the current serving cell as well as the neighbouring cells.
- Extending Traffic Steering (TS) xApp to receive Anomaly Detection messages and trigger predictions request based on these messages.
- KPIMON xApp implementation based on E2SM KPM 2.0.3
- Load prediction capability xAPP will be integrated with the traffic steering use case
- Implementation of Bouncer xApp to will allow the performance benchmarking of the RIC platform (latency of INSERT-CONTROL loop, number of E2 Nodes supported)
- Security xApp – Signalling Storm Protection
- xApp portability SDK to interop with ONF
- Development of Open Test Framework
- O-RU failure and recovery related applications

4.2.8 System Integration

ORAN-SC RIC platform offers complete RIC solutions along with xApps and rApps. Near- RT RIC Platform has been integrated and tested using O-RAN SC based RAN components.

4.3 Comparison SD-RAN vs. ORAN-SC

Criteria	SD-RAN	O-RAN-SC
O-RAN compatibility	• Leverages open interfaces based on O-RAN • TIP has also collaborated with SD-RAN to use ONF's Open Source	• Built on ORAN Specs
Full Software Availability	• SD-RAN v1.1 package is available along with few sample xApps • SDK support for GoLang and Python for developing ML based xAPPs	• Cherry release of O-RAN-SC is available along with RICAPPs package • Supports xApps written in C++, Python and Go
License Type	• Currently it is member only license. Will be made available to public in coming releases	• Apache 2 license
Software Quality	• Well tested code with ONF's RAN simulator • Integrated Airhop's commercial grade eSON as an xAPP	• Tested with Robot based automation platform and O-RAN based RAN solutions
Hardware Requirement	Can be run on any COTS x86 hardware	Can be run on any COTS x86 hardware
Interoperability	• Adheres to O-RAN specifications • xApp interoperability between OSC NR-RIC and ONF µONOS-RIC • ONF µONOS-RIC can interop with O-RAN-SC's non-RT RIC	Adheres to O-RAN specifications
Deployment Considerations	• Release 2.0 is planned to be production-grade software • Flexibility of RAN-in-a-Box (RiaB) deployment option • Trials are done with Airhop's eSON application	Tested with O-RAN-SC based RAN components. Requires hardening to make it commercial grade

Criteria	SD-RAN	O-RAN-SC
Roadmap for O-RAN standards	• Will be upgraded to latest O-RAN specs as and when available	Will be upgraded to latest O-RAN specs as and when available
Community Support	• Led by ONF, which provides various renowned Open-Source networking initiatives such as ONOS, AETHER COMAC, P4, CORD, etc	Formed by collaboration between the O-RAN Alliance and Linux Foundation

5 - Conclusion

5.1 Recommended Open-Source

- For Near-RT RIC, recommendation is to use SW from SD-RAN as more tools are available for testing and also already integrated with Airhop's eSON solution
- For Non-RT RIC, O-RAN provided solution can be used, as currently it is the only available solution

5.2 Minimum Viable Product

Area	Work
5G Testbed	Build open RAN based testbed using SD-RAN in a box PoC setup mentioned in figure 20 and validate 5G call flows without dependency on costly HW Integrating RIC platform to create end-to-end test setup on a COTS HW to validate any RAN optimization algorithms (e.g.: RRM, SON etc.). Details are captured in section 4.1.4.
Minimum Viable Product	Support for Multiple E2 nodes and UEs to add the scalability aspects
	Adding stress testing capability to RAN Simulator to make RIC platform deployment grade
	Porting the O-RAN xApps (e.g., ML approach-based traffic steering xApp) on SD-RAN sdk

Area	Work
Minimum Viable Product	Interoperability of SD-RAN near-RT RIC and O-RAN-SC's non-RT RIC
	Interoperability between xApps from SD-RAN and xApps from O-RAN-SC

5.3 Areas of Future Work

Area	Work
RIC Use cases/xApps	AI/ML based massive MIMO beam forming optimizations based on user distribution and clutter
	Quality of Service (QoS)/Quality of Experience (QoE) predictor using RAN KPIs and traces
	AI/ML assisted Multi User-MIMO UE pairing and Precise Multi User-MIMO pre-coding for better spectral efficiency
	Predict UE scheduling TM modes by means of machine learning techniques and application information
	Dynamic spectrum sharing for 4G-5G co-existence
	AI/ML assisted interference detection and mitigation, Address cell overshoot issues, Minimize coverage gaps and avoid atmospheric ducting phenomenon
	AI/ML based network Congestion prediction by monitoring user plane KPIs and notify/recommend possible actions to the operator.

6. References

- O-RAN.WG3.RICARCH-v01.01 available for download at (https://oranalliance.atlassian.net/wiki/download/attachments/207421446/O-RAN.WG3.RICARCH-v01.01.docx?api=v2)

- O-RAN.WG2.Non-RT-RIC-ARCH-TR-v01.01 available for download at https://oranalliance.atlassian.net/wiki/download/attachments/507740285/O-RAN.WG2.Non-RT-RIC-ARCH-TR-v01.01.pdf?api=v2

- O-RAN Alliance White Paper, Feb 2020 "O-RAN Use Cases and Deployment Scenarios Towards Open and Smart RAN" available for download at https://static1.squarespace.com/static/5ad774cce74940d7115044b0/t/5e95a0a306c6ab2d1cbca4d3/1586864301196/O-RAN+Use+Cases+and+Deployment+Scenarios+Whitepaper+February+2020.pdf

- Telefónica views on the design, architecture, and technology of 4G/5G Open RAN networks available for download at https://www.telefonica.com/documents/737979/145981257/Whitepaper-OpenRAN- Telefonica.pdf/3a160ca9-c325-a3d6-a6da-f9453616144dhttps://opennetworking.org/uncategorized/onf-commits-to-supporting-o-ran-alliance/

- https://wiki.o-ran-sc.org/display/ORAN/Projects

- https://static.sched.com/hosted_files/ones2020/99/ONeS_NA_2020_Thoralf_Czichy__Matti_Hiltunen.pdf

- https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=10715420

- https://opennetworking.org/sd-ran/

- https://opennetworking.org/wp-content/uploads/2021/04/SD-RAN-v1.1-Techinar-Slides.pdf

- https://opennetworking.org/uncategorized/onf-commits-to-supporting-o-ran-alliance/

- https://www.etsi.org/deliver/etsi_gs/mec/001_099/002/02.01.01_60/gs_mec002v0201_01p.pdf

tsdsi
India's Telecom SDO

Telecommunications
Standards Development
Society, India

TSDSI is an autonomous, membership based, Standards Development Organization (SDO) for Telecom/ICT products and services in India.

- Room No IIA 304, Second Floor,
Bharti School of Telecommunication Technology & Management IIT Delhi,
Hauz Khas, New Delhi – 110016, India

- secretariat@tsdsi.in

- https://tsdsi.in/

- https://www.linkedin.com/company/tsdsi/

- https://twitter.com/TSDSI_India