M-CORD test architecture and roadmap

This document is a proposal for M-CORD test infrastructure. The goal is to build a unified test framework to test various M-CORD components (e.g. SD-RAN, Split RAN, Fronthaul, PGW, SGW, MME, eNodeB, HHS, etc ). The document includes a few test setups and test cases as example, but it is not designed to capture all test cases and test setups.

## Objective of M-CORD test environment

The primary objectives of test architecture are:

1. Support end-to-end test as well as subsystem test - The system should be able to support End-to-end testing ( UE-eNodeB-EPC-Internet ) as well as subsystem test (e.g. SD-RAN, PGW/SGW only or eNodeB only ). For subsystem testing, the framework should be able to extenicate a test environment to “wrap around” the DUT ( device under test ). The following diagrams shows an example of end-to-end test environment. The framework is able to setup the LTE network and generate test traffic.

DUT

UE

RRH

eNB

SGW

PGW

MME

Test Framework

Here is another example of test environment for core network testing. To perform test, we will need to create an emulated eNodeB to interface to MME ( via S1-AP ) and S-GW ( via S1-U). The test framework is able to generate appropriate protocol sequences to execute desired test cases.

DUT

eNB emulation

SGW

PGW

MME

Test Framework

The eNodeB emulation could be a VM/container running in one of the compute nodes inside the pod or a separate piece of test equipment connect to the network interface of the pod. The test framework should be able to configure and execute commands to control the emulated elements.

1. Portability of test cases across multiple test setups - The test cases and test scripts should be independent of the test setups. For example, if we have multiple M-CORD pods with different names and IP addresses, the script should be able to read XOS model data of the pod to know the names, IP addresses, and elements inside the Pod and execute the test cases.
2. Reusability of test cases for different testbed - Similarly, the test cases should be able to run on full end-to-end test as well as subsystem test. For example, the core network throughput test should work with full setup or just S/PGW as DUT.
3. Common framework across different CORD projects - ideally, the test framework should be the same across M-CORD, R-CORD, and E-CORD to minimize duplicated effort.
4. Support of multiple test equipment vendors - Due to the complexity of 3GPP standard, we are likely to have 3rd party vendors such as Spirent, IXIA, NG4t to provide test equipments or test VMs for M-CORD testing. We should try to work with test equipment collaborators to create common northbound interfaces so the test cases can be equipment-independent[[1]](#footnote-1). The architecture would look like the following:

System under Test

Test equipment

Test equipment

Test driver

Test driver

Config pusher

Framework

Test cases

Testbed model

Test result

Data collector

## Roadmap

### Phase 1 - framework development for PoC

The goal for first phase is to develop and bring up initial framework as Proof of Concept. Ideally, we can work with different collaborators to construct the test framework for a testbed ( wither CiaB or full Pod) and demonstrate functionality.Phase 2 - smoke-test suite for build and release

To support continuous development and release, we should construct a small set of test cases as sanity test for each release. This will help stabilize the code base to support future development. The ultimate goal should be to enable CI/CD model where injection of new features can automatically be validated by reports such as “SONARQUBE” for feedback loop to develops when the builds are broken or regression tests fail.

Phase 3 - Functional & performance test suite

Building full feature M-CORD test suite will help improve software quality and drive the system towards production readiness. Ultimately, the ultimate goal for performance benchmarks are in hands of independent system test collaborators and partners. They will be in charge of independent reports for each major release or application and their test bed environment such as Hardware and acceleration resources. The service providers operation will be the end users of these independent reports to help them make deployment decisions based on open CORD platforms

## Collaboration with test equipment vendors

There are couple areas we can use the help from various test equipment collaborators:

1. In phase 1, we would like to get help extend framework to support M-CORD use case. We also like to have at least 1 test equipment integrated into the framework to prove the correctness of design.
2. In phase 2, we can leverage difference collaborators to integrate their equipments into M-CORD testbed and have them contribute test drivers ( for their test equipment) and test cases (can be used for all equipment vendors)
3. In phase 3. We can use help from collaborators to construct specific testbed and demonstrate the performance of the system.

## Common architecture with R-CORD

The following is a diagram I found on [Wiki](https://wiki.opencord.org/display/CORD/Cord-Tester%2BDesign%2BDocs). The test container includes both test framework and test modules ( test cases?).



### **What to Test**

Testing will cover the following categories.

* Feature/Functionality Testing
* Performance Testing
* Resiliency Testing
* Security Testing

### **How to Test**

In order to pass or fail a test case, data needs to be collected/measurements need to be made. Following are the possible dimensions along which these activities can take place.

* Messages that are flowing among the nodes (virtual and/or physical) and the applications associated with the nodes
* Information collection/measurements by logging into the nodes and applications
* Information collection/measurements that are carried out at entities connected with the nodes and applications. Examples of such entities are vendor-provided or generic managers (EMS, VNFM etc.), testing/assurance applications etc.
* Information collection/measurements at the machine level (virtual and/or physical)
* Information collection/measurements outside of M-CORD. For example, at the end-device, destination-servers etc.

## **Checklist for Test Plan Creation**

* Feature/Functionality Testing
	+ Can we list the use cases? Do we have message flow diagrams associated with the use cases?
	+ Can we create the high level test cases? What do we want to test for each use case?
	+ Can we fill in the test details? What to check? How to collect the information needed for the checking? What criteria to use to pass/fail the TC?
* Performance Testing
	+ Do we have the information to build upon the feature/functionality test cases, and convert them into performance tests cases? Do we have a traffic model for traffic generation?
	+ Can we fill in the test details? What stats/KPIs to check? How to collect the stats/KPIs? What criteria to use to pass/fail the TC?
* Resiliency Testing
	+ Do we have the information related to redundancies being used in the M-CORD system? Impairments that can impact the M-CORD system?
	+ Can we fill in the test details? What to check? How to collect the information needed for the checking? What criteria to use to pass/fail the TC?
* Security Testing
	+ Do we have a list of potential threats associated with the M-CORD system?
	+ Can we convert the threats into test cases?
	+ Can we fill in the test details? What to check? How to collect the information needed for the checking? What criteria to use to pass/fail the TC?

### **When to Test**

For test planning purposes, this document views M-CORD in the following manner.

* M-CORD’s base offering is a service-friendly mobile cellular network. The key pillars of this base are OpenStack, ONOS, and XOS.
* On top of this base, value added services (such as slicing) are developed and deployed

Based on the above, following two SUTs can be identified.

1. Base Mobile Cellular Network
	1. Setup the Mobile Cellular Network
		1. The RAN that comes up via XOS, will be inherently sliced i.e. we will bring up a Mobile Cellular Network slice
		2. This slice will have a RRU, BBU, MME, SGW, and PGW
	2. E2E scenarios can be tested with this SUT
		1. Overall Focus
			1. Attach/Handover
		2. RAN Focus
			1. Scheduling Efficiencies
			2. Interference / Impact
			3. LTE capability set
			4. Front Haul option
		3. CN Focus
			1. Authentication / Authorization
			2. Accounting/Charging
			3. QOS
			4. Policy Enforcement
			5. Traffic steering
			6. Content Optimization
			7. Support for Broadcast and Multicast
2. Network with Value Added Services (such as slicing)
	1. Slice the RAN based on the profiles defined via ProgRAN
		1. The profile based slices introduce a second slicing dimension in the RAN, which is in addition to the above-mentioned XOS slicing.
		2. The profile based slicing contains RAN specific parameters such as handover management etc.
	2. Stich the RAN slice with the CN slice
		1. The CN slice is assumed to be a single-dimension slice that was setup via XOS i.e. unlike the RAN, there is no additional profile based slicing dimension in the CN.
	3. Slicing scenarios can be tested via this SUT. Some possible scenario dimensions.
		1. Slice categorization options
			1. Services ( IOT, AR/VR, Video, Data, Gaming, Signaling)
			2. User type ( premium, gold, silver)
			3. Device Types (static IOT devices, Smart UEs, Vehicular, high mobility deices)
			4. Application / data type (Voice, real time video, Real Time Data , Best Efforts, Time shifted)
			5. QOS, QOE policies ( Application centric, Service Centric, User specified)
			6. Enterprise ( financial, health, education, etc.)
			7. Location based , Delivery Path Isolation
		2. Slice Control utilizing CORD Control functions •Slice Management
			1. Slice Elasticity Options
			2. Common analytics control APIs
		3. Cross layer slice management options at PHY, MAC and RRC layers
1. This will need more detail feasibility analysis [↑](#footnote-ref-1)