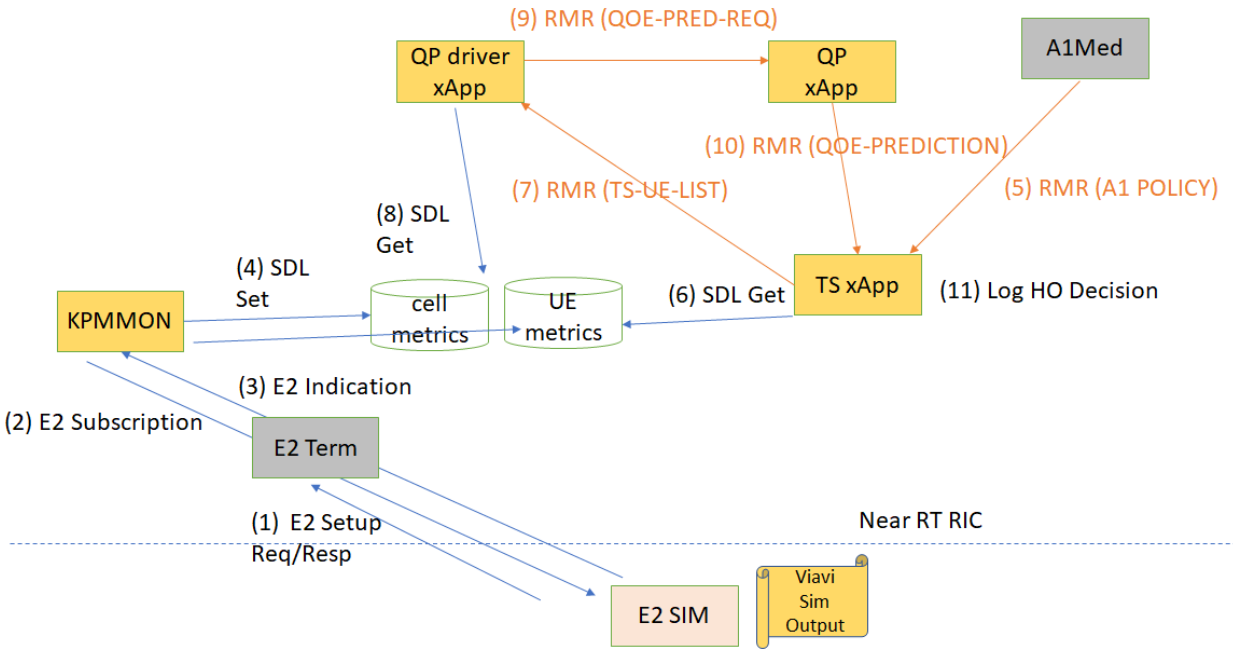
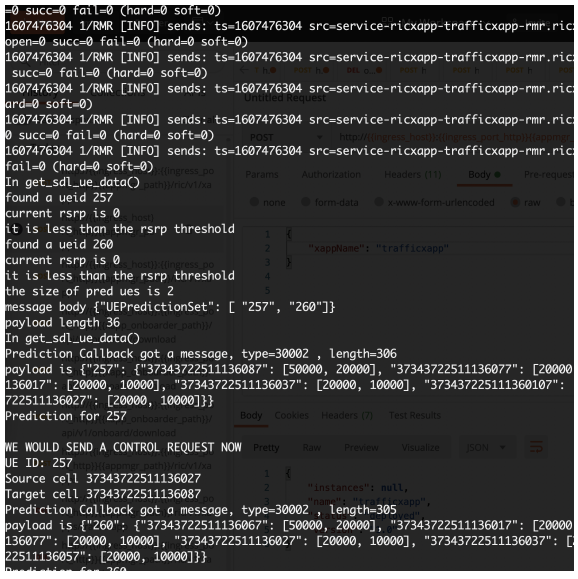
Traffic Steering Flows



		OTF	OAM	NONRTRIC	RICP	RICAPP	O-DU	O-CU	Test Result	Notes
Deployment Artifacts	Docker Container									E2sim docker image needs to be generated locally for different RIC instance, wil KPIMON needs CI jobs for image releasing QPdriver image 1.0.9 in staging is different than the 1.0.9 image in release repo Need to release QP and TS xApp images
	xApp Descriptors									Need to update TS, QP, QP driver descriptors to release version Need to create a descriptor for KPMMON
E2E Flows	E2 setup									
	E2 Subscription									
	E2 Indication									
	KPM MON write to SDL									
	Create A1 Policy									
	TS read from SDL									
	TS sends triggers to QP driver									
	QP driver gets data from SDL									



	QP driver send data to QP								
	QP provide prediction								
	TS make control decision								

MWC LA Demo Video

Your browser does not support the HTML5 video element

Test Manual

Deploy E2Simulator

There are several steps below that need to be taken to run this successfully. Sorry it is not yet as smooth as it should be. Some of these need to be automated properly, and some have to do with platform issues. If someone could take some of these on it would be helpful.

1. Deploy RIC platform

2. kubectl rollout restart deployment --namespace ricplt deployment-ricplt-e2mgr deployment-ricplt-e2term-alpha (The e2mgr and e2term need to be restarted; this is due to some issues with health check; I don't know whether it has been resolved by that team)

3. clone sim/e2-interface

4. In the root directory of e2sim: Follow directions in README to produce the deb files

5. Since the deb files are not yet pushed to package cloud, we need to copy them

6. cp e2sim*deb ../e2sm_examples/kpm_e2sm

7. cd ../e2sm_examples/kpm_e2sm

8. Edit the Dockerfile at the bottom to have IP address of service-ricplt-e2term-sctp-alpha service

9. docker build .

10. docker tag <tag just built> e2simul:0.0.2

11. helm install –namespace ricplt helm

Once deployed, kubectl logs will show:

1. An E2 Setup Request from E2 Simulator to E2 Term
2. An E2 Setup Response from E2 Term to E2 Simulator

Onboarding and Deployment of xApps

The Use case involves four different xApps:

- Traffic Steering (TS) xApp (AT&T)
- QoE Prediction Driver (QPDriver) xApp (AT&T)
- QoE Prediction (QP) xApp (AT&T)
- KPIMON xApp (Samsung)

The first step is to deploy KPIMON xApp

Since KPIMON xApp does not yet have a CI job to create an image in Nexus and also does not have an xApp descriptor,

currently KPIMON deployment must be done in the following way:

1. clone scp/ric-app/kpimon **(note it is different than the ric-app/kpimon repo)**
2. docker build .
3. docker tag <image tag just created> nexus3.o-ran-sc.org:10002/ric-app-kpimon:1.0.0
4. curl -X POST --data-binary @xappkpimon-0.2.0.tgz http://<vm-name>:32080/helmrepo/api/charts **(Push hand-crafted helm chart to appmgr chart museum)**

We include the helm chart tarball here:

Now deploy the other three xApps

Each of these three xApps have a descriptor in their gerrit repo under the xapp-descriptor/ directory.

~~None of them have xapp specific controls and therefore no individual json schema~~

(Update Nov 23, QPDriver needs a separate schema file)

Here are the URLs for each which can be included in HTTP POST call to onboard tool

TS xApp:

https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/ts.git;a=blob_plain;f=xapp-descriptor/config.json;hb=refs/heads/master

QPDriver xApp:

Descriptor:

https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/qp-driver.git;a=blob_plain;f=xapp-descriptor/config.json;hb=0a0dff549b933f049b05fc26ce2f0a13da853b78

Schema:

https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/qp-driver.git;a=blob_plain;f=xapp-descriptor/controls.json;hb=refs/heads/master

QP xApp

https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/qp.git;a=blob_plain;f=xapp-descriptor/config.json;hb=HEAD

Here is the URL for the Xapp Onboarder in your environment. The values 'ingress_host', 'ingress_port_http' and 'xapp_onboarder_path' refer to the hostname and port for reaching Kong ingress controller, and the ingress path assigned to xapp onboarder (likely the path is set to 'onboard').

http://{{ingress_host}}:{{ingress_port_http}}/{{xapp_onboarder_path}}/api/v1/onboard/download

As an example, the message body for the TS xapp is below. This needs to be sent in a POST request with Content-Type equal to 'application/json'

```
{
      "config-file.json_url": "https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/qp-driver.git;a=blob_plain;f=xapp-descriptor/config.json;hb=HEAD"
}
```

To subsequently deploy any of these xapps, use the following command:

curl --location --request POST "http://{{ingress_host}}:{{ingress_port_http}}/{{xapp_onboarder_path}}/ric/v1/xapps" --header 'Content-Type: application/json' --data-raw '{"xappName": "trafficxapp"}'

(where xappName would be later set to 'qpdriver' and 'qp' to deploy the QP Driver and QP xapps respectively)

Creation of Traffic Steering Policy Type and Policy Instance

TS xApp consumes an A1 Policy

Below are the steps for providing it with a policy.

Policy Type Create

Copy the following into a file called create.json -
Policy Type Create

```
{"name" : "tsapolicy", "description" : "tsa parameters", "policy_type_id" : 20008, "create_schema" : { "$schema" : "
http://json-schema.org/draft-07/schema#", "type" : "object", "properties" : { "threshold" : { "type" : "integer",
"default" : 0 } }, "additionalProperties" : false } }
```

Then run:

curl -X PUT --header "Content-Type: application/json" --data-binary @create.json http://<Base URL for Kong>/a1mediator/a1-p/policytypes/20008

Policy Create

Run command:

curl -X PUT --header "Content-Type: application/json" --data "{\"threshold\" : 5}" http://<Base URL for Kong>/a1mediator/a1-p/policytypes/20008/policies/tsapolicy145