
# FlexRIC tutorial: xApp development

OPEN AIR INTERFACE

**Mikel Irazabal**

**OSA Senior Software Engineer**

**31/01/23**

# 0 Outline

# 1 Outline

# 1 O-RAN Architecture



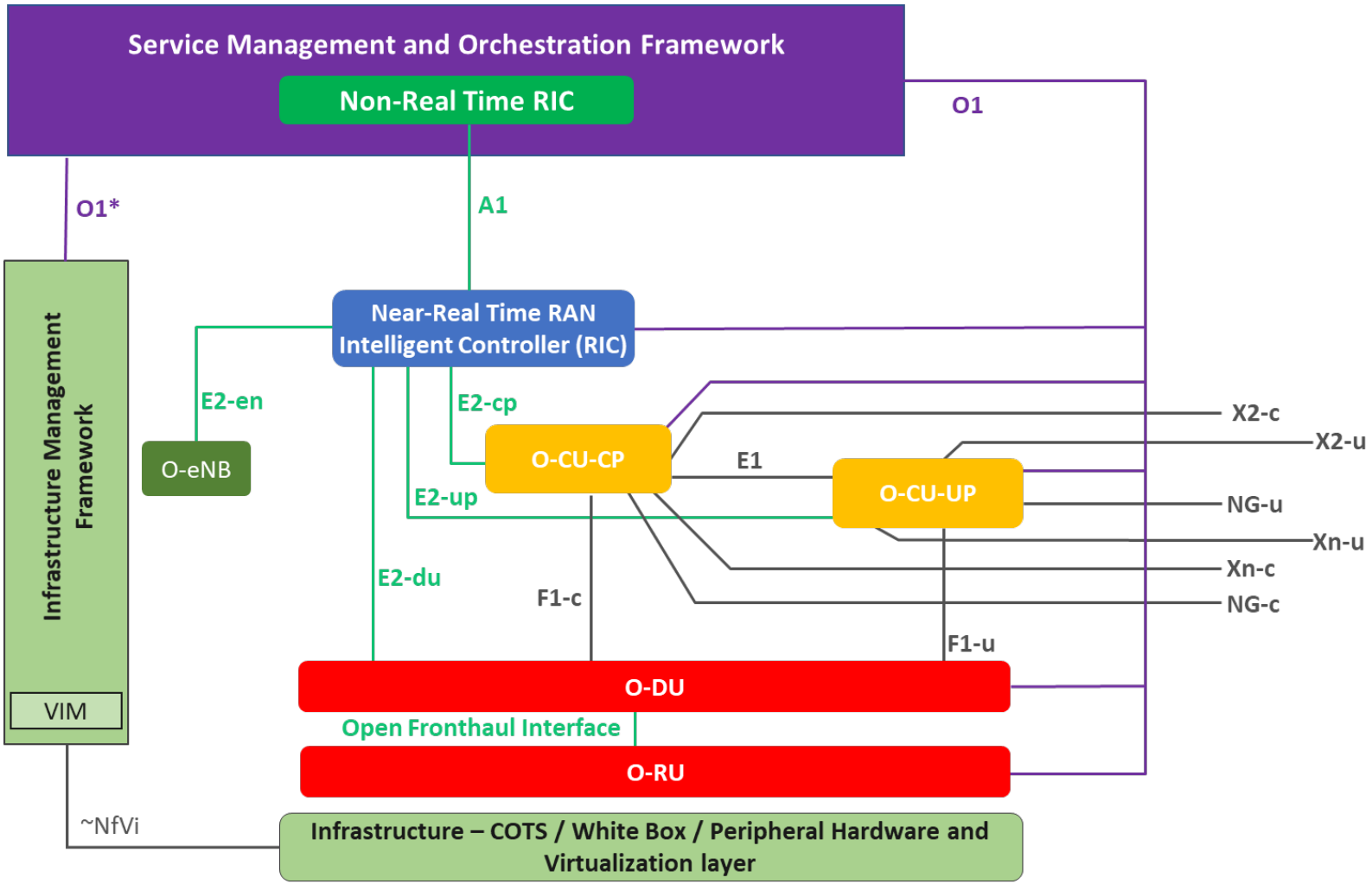



Figure: O-RAN Architecture

# 2 Outline

- Principles
  - Ultra-lean for low-latency or resource-restricted use cases i.e., follows the zero-overhead principle
  - Flexibility and forward compatibility towards novel use cases through the utilization of static and dynamic polymorphism
  - A RAT-agnostic and vendor neutral SD-RAN design (e.g., OAI )
  - Service Models (SMs) are implemented as shared objects to reduce the coupling

# 2 FlexRIC v0.1

- E2AP protocol v1 (20/26 messages in ASN.1)
- MAC,RLC and PDCP Service Models to monitor
- nearRT-RIC and E2 Agent
- iApps for low-latency communication
- Approx. 10 K LOC
- FlexRIC paper https://bit.ly/3uOXuCV

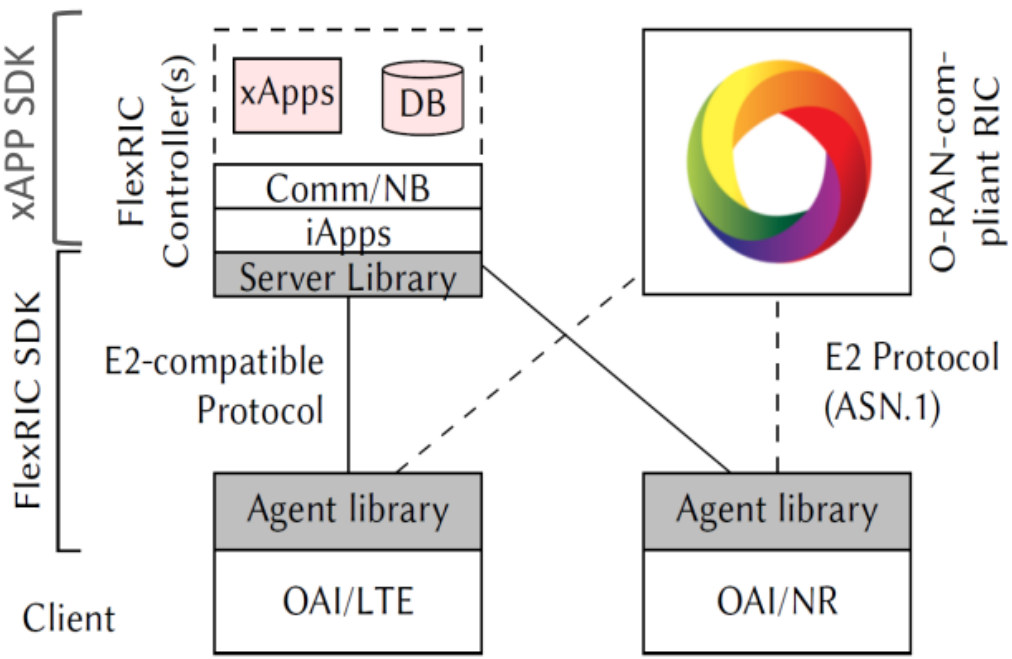



Figure: FlexRIC v0.1

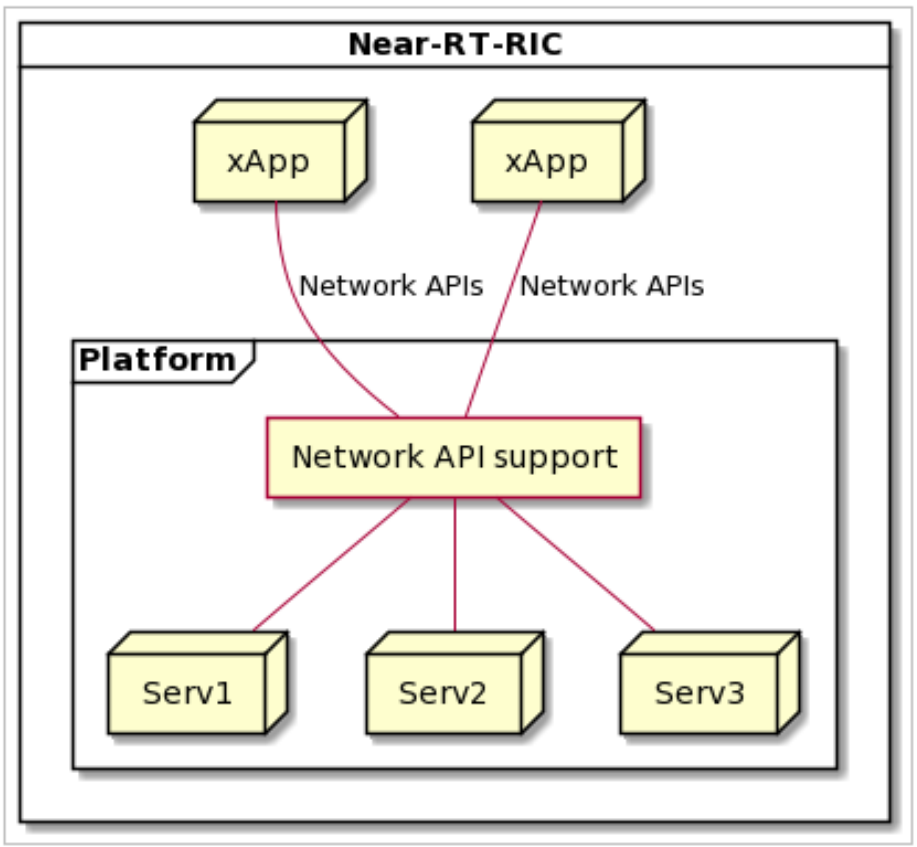
# 2 FlexRIC v0.2

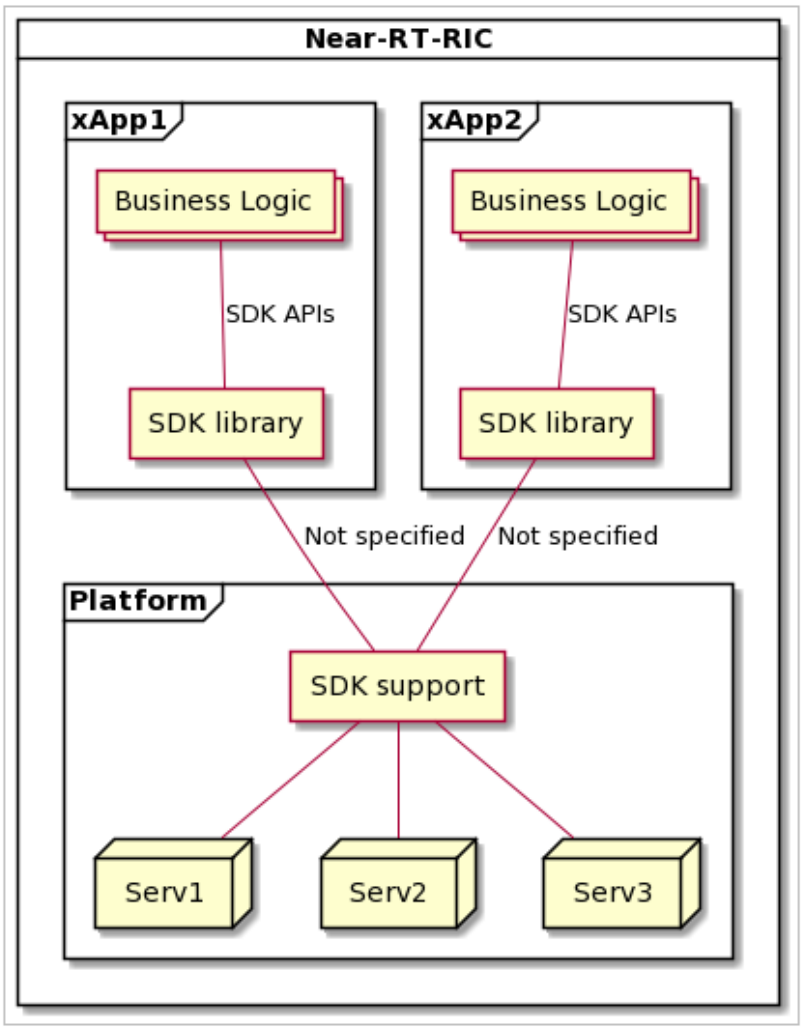
- FlexRIC has become Multi-X
  - Multi-vendor i.e., OAI and srsRAN
  - Multi-RAT i.e., 4G and 5G
  - Multi-language i.e., C/C++ and Python
  - Multi-agent
  - Multi-xApp
- New protocol between the xApp and the RIC i.e., E42
- New SMs have been developed (Slice and Traffic Control)
- FlexRIC has grown from approx. 10K to 50K LOC.

# 3 Outline

# 3 Network API vs SDK approach





# 3 Network API vs SDK approach

- Network API approach: Each relevant Near-RT RIC endpoint exposes a network endpoint and specifies a particular data encoding protocol and a network transport protocol that should be used to communicate with it. A different network API may be specified for each Near-RT RIC API reflecting a trade-off between different service requirements.
  - An application layer protocol used to carry a set of messages which normally contain multiple Information Elements (IEs);
  - A data encoding protocol (ASN.1, Protobuf, JSON, etc.);
  - A network transport protocol (SCTP, HTTPS, gRPC, etc.);
  - Associated security and encryption methods.
- SDK approach: The Near-RT RIC vendor provides an SDK (software development kit). This SDK is a software library which handles all connection management and exposes a simple API for the xApp to interact with the Near-RT RIC. The interface between the SDK library embedded in the xApp and the Near-RT RIC Platform may be either vendor proprietary or aligned to the specified Network API.
  - Providing simple APIs to trigger commonly used functionality;
  - Handling routine management tasks “under the hood”;
  - Providing tools for debugging, building, testing applications.

# 3 O-RAN xApp Requirements I

- xApp may enhance the RRM capabilities of Near-RT RIC;
- xApp may be associated with zero, one or more E2SMs;
- xApp shall use Near-RT RIC APIs to make use of the Information Elements (IEs) of E2SMs that are associated with it;
- xApp that is associated with a given E2SM shall be able to interface with any E2 Node that supports that E2SM without any intermediary xApp;
- xApp shall be able to receive event-triggered information on RAN information and time-varying network state.
- xApp shall provide collected logging, tracing and metrics information to Near-RT RIC;

# 3 O-RAN xApp Requirements II

- xApp shall provide a descriptor that includes the following basic information of the xApp:
  - Configuration: It includes a data dictionary for configuration data, i.e., meta data such as a YANG definition or a list of configuration parameters and their semantics. It may also include an initial configuration of xApp;
  - Control: It includes the types of data that an xApp consumes and generates, in order to perform control capabilities (e.g., xApp URL, parameters, input/output type);
  - Metrics: It includes 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 management and orchestration of the xApp, aligned with [4];
- xApps shall communicate with Near-RT RIC platform via Near-RT RIC APIs;
- xApp shall register the Near-RT RIC APIs it produces;
- xApp shall be capable of discovering the Near-RT RIC APIs they consume.

# 3 O-RAN Architecture xApp APIs

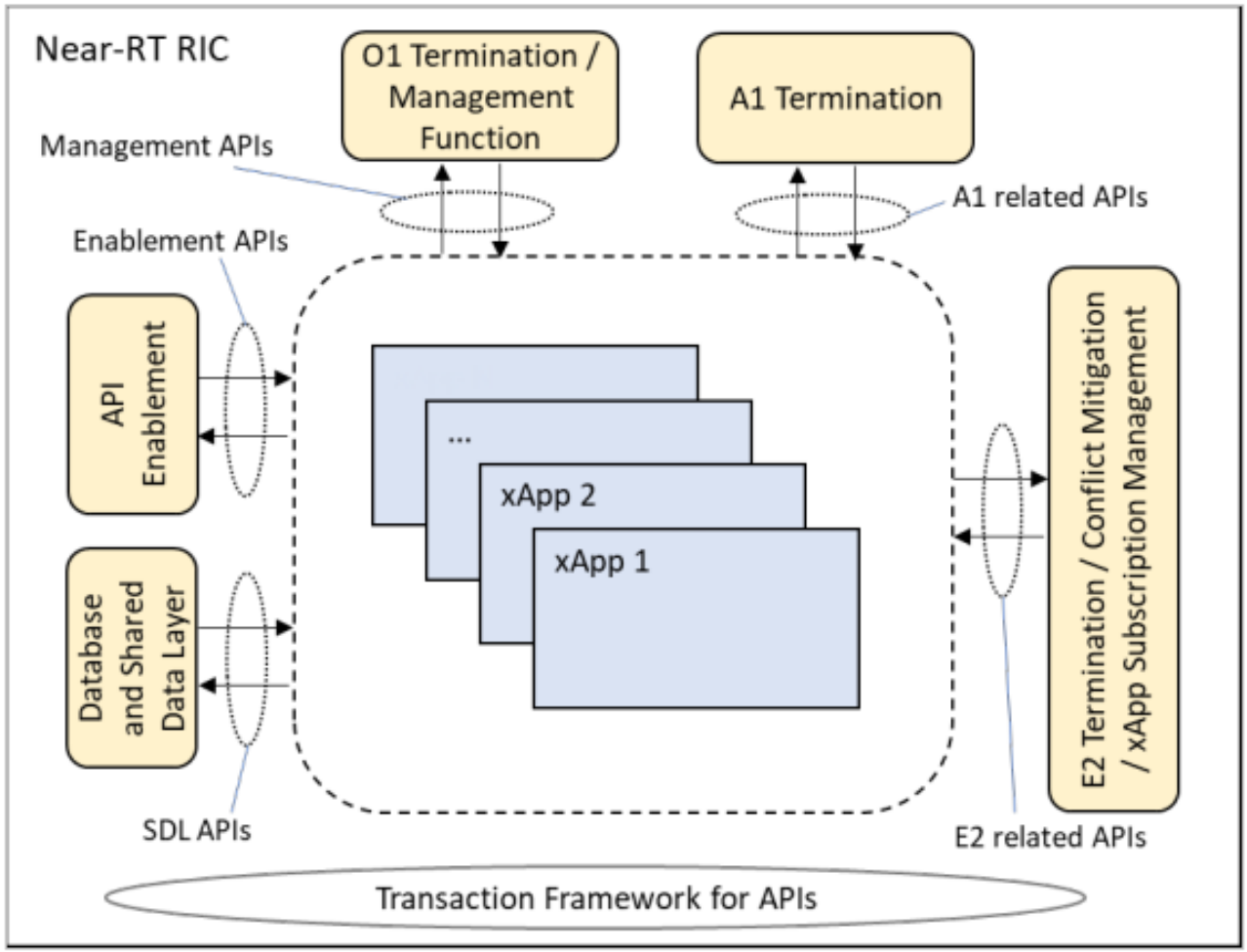



Figure: Overview of Near-RT RIC APIs

- A1 APIs allowing to access A1 related functionality;
- E2 APIs allowing to access E2 related functionality and associated xApp Subscription Management and Conflict Mitigation functionality;
- O1 Management APIs allowing to access management related functionality;
- SDL APIs allowing to access Shared Data Layer related functionality;
- Enablement APIs between xApps and API enablement functionality.

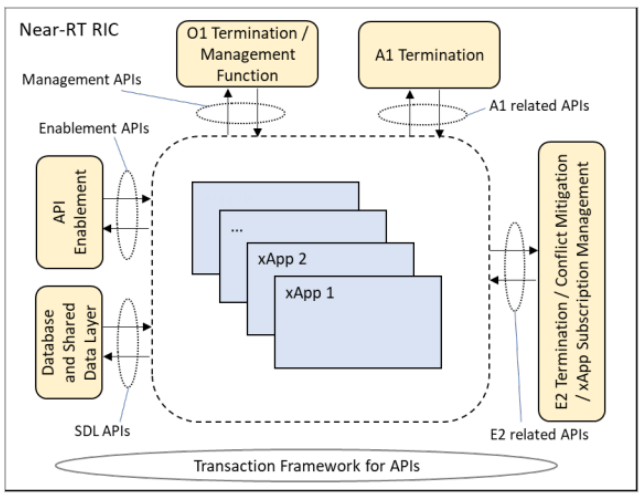



Figure: Overview of Near-RT RIC APIs

# 4 Outline

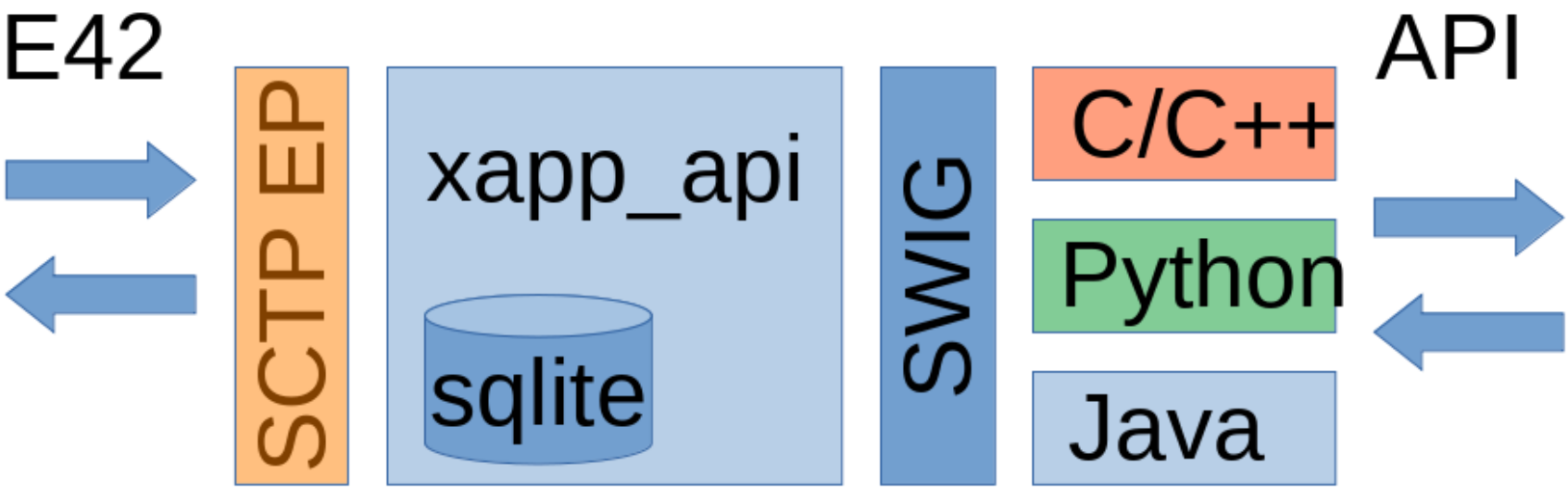



Figure: Overview of FlexRIC's SDK

- O-RAN has some limitations
  - A protocol needed between the SDK and the nearRT-RIC
  - Need for an E2 Node concept
  - Not defined the semantics of *on-boarding*
- Therefore, we decided to enhance E2 with E42 with 5 new messages i.e., E42 Setup Request, E42 Setup Response, E42 Subscription Request, E42 Subscription Delete Request and E42 RIC Control Request,

- E42 Setup Request: available RAN Functions of the SDK. No Global ID info as E2 Setup Request
- E42 Setup Response: Connected E2 Nodes with accepted RAN Functions and xApp ID

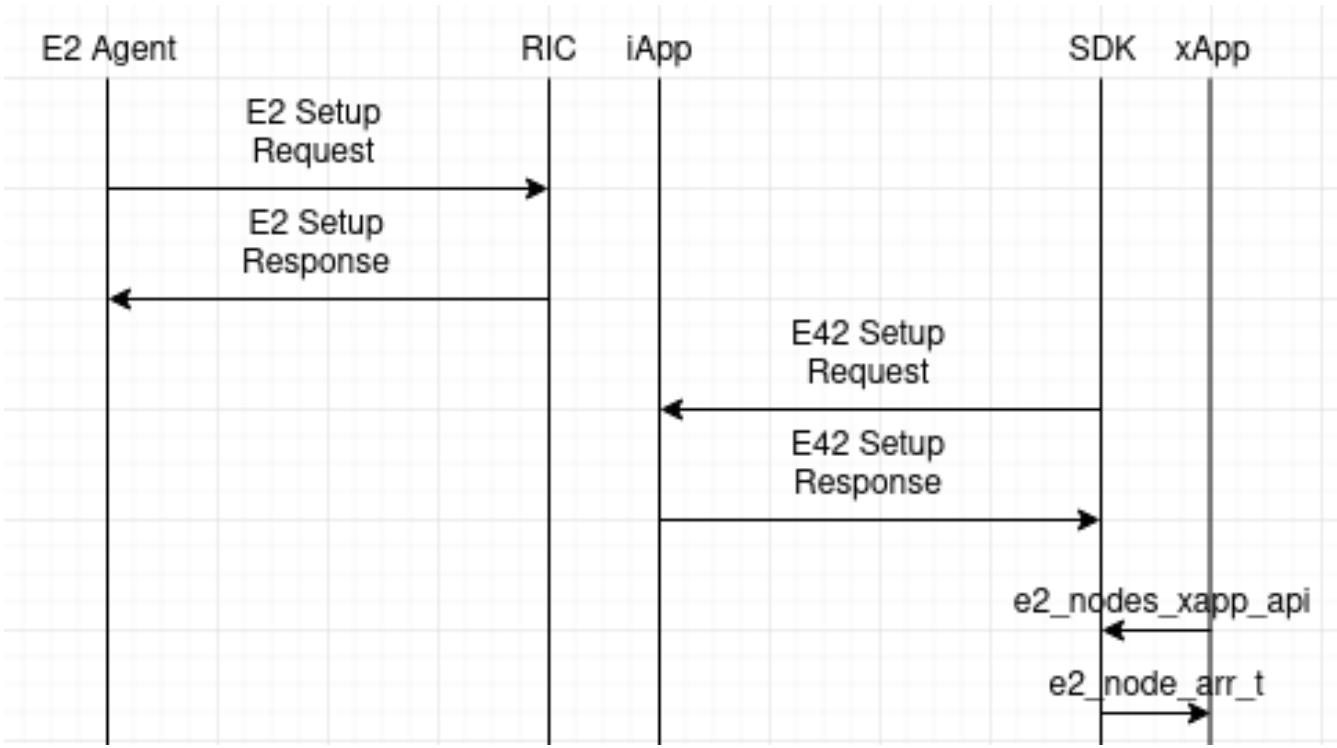
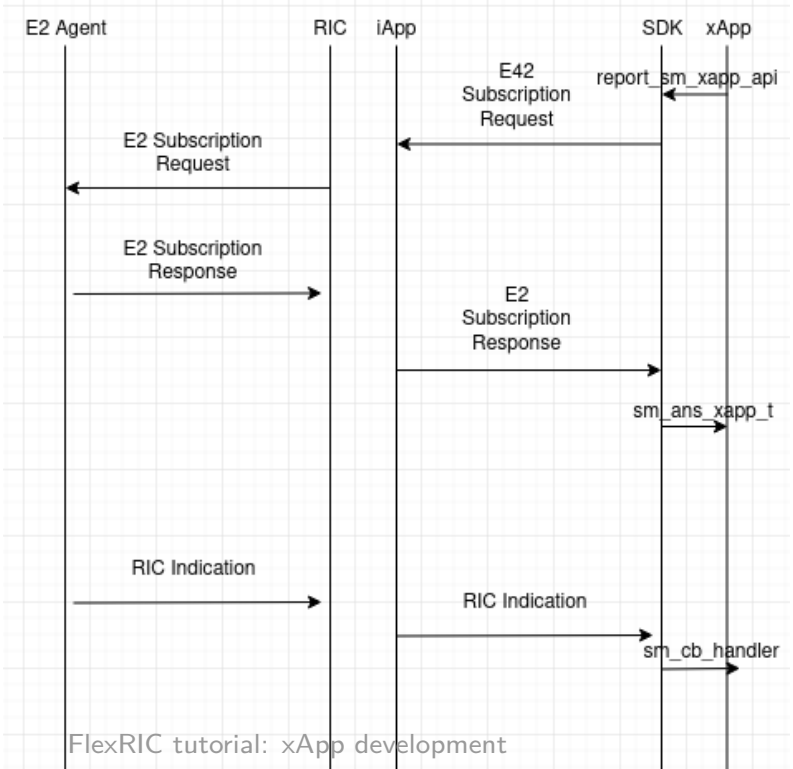



Figure: E42 Initial connection

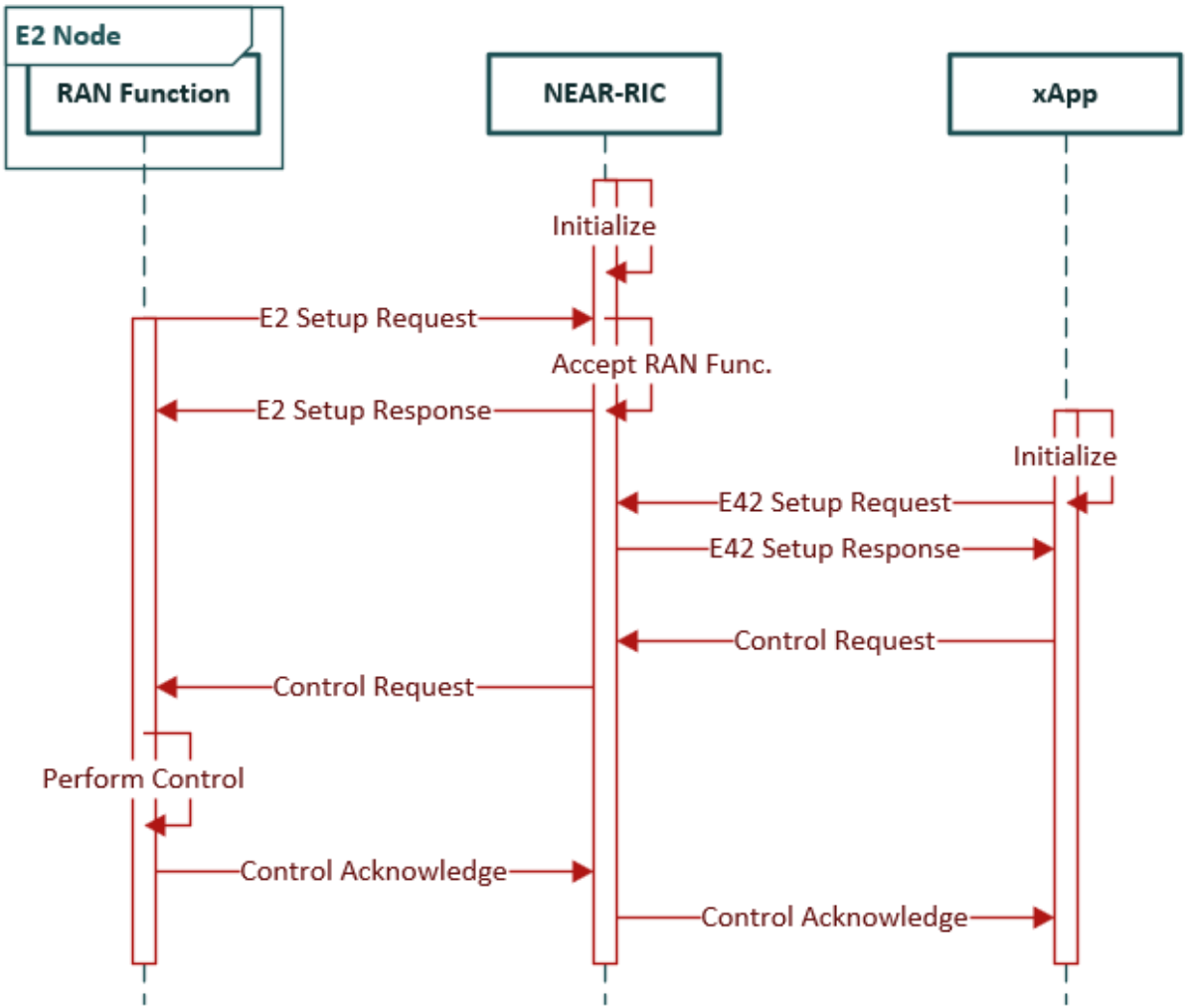
# 4 E42 Subscription Request

- report_sm_xapp_api: E2 Node ID, SM ID, interval, callback
- E42 Subscription Request: xApp ID, E2 Node ID E2 Subscription Request
- sm_ans_xapp_t: Handle for the user. Needed to remove the subscription request
- sm_cb_handler: Function where the results arrive in the xApp code. The xApp, also writes per default the data into an SQLite DB



# 4 E42 Control

- Report_sm_xapp_api: E2 Node ID, SM ID, callback interval, E42 Subscription Request: xApp ID, E2 Node ID E2



- Message handler, message generator, thread safe queue, pending events, plug-in, sync User Interface and active procedures

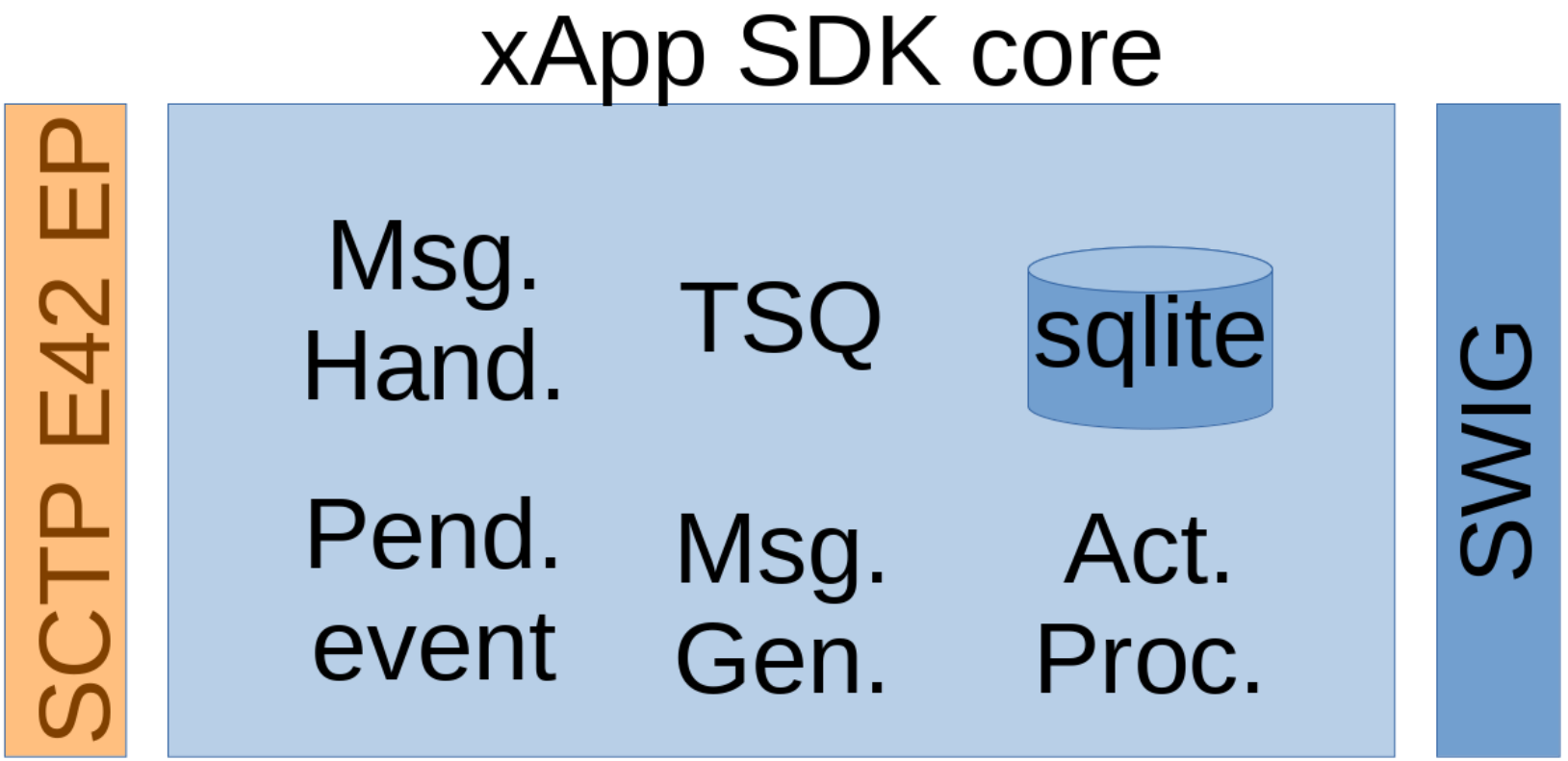



Figure: xApp SDK core

# 4 Recently found data race in E42 Control

- The message came back from the E2 Agent, while it associated pending event was not stored

```
Thread 5 "xapp_slice_moni" received signal SIGABRT, Aborted.
[Switching to Thread 0x7ffff0bfe700 (LWP 27247)]
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
51      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
(gdb) bt
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
#1  0x00007ffff63357f1 in __GI_abort () at abort.c:79
#2  0x00007ffff63253fa in __assert_fail_base (fmt=0x7ffff64ac6c0 "%s%s%s:%u: %s%sAssertion `%s' failed.\n%n",
    assertion=assertion@entry=0x5555557a12a8 "z_node != tree->dummy && \"Trying to extract a key not found in the tree\"",
    file=file@entry=0x5555557a1060 "/home/orandesktop/project/flexric/src/util/alg_ds/ds/assoc_container/assoc_rb_tree.c",
    line=line@entry=418, function=function@entry=0x5555557a1480 <__PRETTY_FUNCTION__.3055> "assoc_rb_tree_extract") at assert.c:92
#3  0x00007ffff6325472 in __GI___assert_fail (
    assertion=0x5555557a12a8 "z_node != tree->dummy && \"Trying to extract a key not found in the tree\"",
    file=0x5555557a1060 "/home/orandesktop/project/flexric/src/util/alg_ds/ds/assoc_container/assoc_rb_tree.c", line=418,
    function=0x5555557a1480 <__PRETTY_FUNCTION__.3055> "assoc_rb_tree_extract") at assert.c:101
#4  0x000055555559a6e8 in assoc_rb_tree_extract (tree=0x7b7000000668, key=0x7ffff0bbdf70)
    at /home/orandesktop/project/flexric/src/util/alg_ds/ds/assoc_container/assoc_rb_tree.c:418
#5  0x00005555555a90a4 in bi_map_extract_right (map=0x7b7000000638, key2=0x7ffff0bbdf70, key2_sz=16)
    at /home/orandesktop/project/flexric/src/util/alg_ds/ds/assoc_container/bimap.c:107
#6  0x00005555555a4ccb in rm_pending_event_ev (ev=0x7ffff0bbdf70) at /home/orandesktop/project/flexric/src/xApp/pending_event_xapp.c:
#7  0x0000555555621f32 in rm_pending_event_xapp (xapp=0x7b7000000000, ev=0x7ffff0bbdf70)
    at /home/orandesktop/project/flexric/src/xApp/msg_handler_xapp.c:88
#8  0x0000555555623529 in e2ap_handle_control_ack_xapp (xapp=0x7b7000000000, msg=0x7ffff0bbe160)
    at /home/orandesktop/project/flexric/src/xApp/msg_handler_xapp.c:300
#9  0x00005555556223c4 in e2ap_msg_handle_xapp (xapp=0x7b7000000000, msg=0x7ffff0bbe160)
    at /home/orandesktop/project/flexric/src/xApp/msg_handler_xapp.c:133
#10 0x000055555559f773 in e2_event_loop_xapp () at /home/orandesktop/project/flexric/src/xApp/e42_xapp.c:237
#11 0x000055555559fd01 in start_e42_xapp (xapp=0x7b7000000000) at /home/orandesktop/project/flexric/src/xApp/e42_xapp.c:282
---Type <return> to continue, or q <return> to quit---f7
#12 0x00005555555952fd in static_start_xapp (a=0x0) at /home/orandesktop/project/flexric/src/xApp/e42_xapp_api.c:56
#13 0x00007ffff6d356ae in ?? () from /usr/lib/x86_64-linux-gnu/libtsan.so.0
#14 0x00007ffff66ed6db in start_thread (arg=0x7ffff0bfe700) at pthread_create.c:463
#15 0x00007ffff641661f in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95
```

Figure: Diagram

# 4 xApp SDK design principles

- Simplicity. KISS principle
- Multi-language i.e., C/C++, Python, golang, java ...
- Efficient

- xApp API
  - Function init and stop functions
  - Function to get the E2 Nodes
  - Function to subscribe and remove subscription
  - Function to control

```
void init_xapp_api(fr_args_t const*);

bool try_stop_xapp_api(void);

e2_node_arr_t e2_nodes_xapp_api(void);

// returns a handle
sm_ans_xapp_t report_sm_xapp_api (global_e2_node_id_t* id, uint32_t ran_func_id, inter_xapp_e i, sm_cb handler);

// Remove the handle previously returned
void rm_report_sm_xapp_api(int const handle);

// Send control message
sm_ans_xapp_t control_sm_xapp_api(global_e2_node_id_t* id, uint32_t ran_func_id, sm_ag_if_wr_t const* wr);
```

Figure: xApp API functions

- xApp queries: Since some more complex operation may be required by the user, we decided to embedded a DB into the xApp, so that queries can be made directly to a DB, rather than maintaining the states in the code

- SWIG is an interface compiler that connects programs written in C and C++ with scripting languages such as Perl, Python, Ruby, and Tcl. (Java, JavaScript, Go, D, C sharp ...)
- The idea is that the SDK is written in efficient C and that according to our needs, we use the SWIG compiler to generate code for other languages

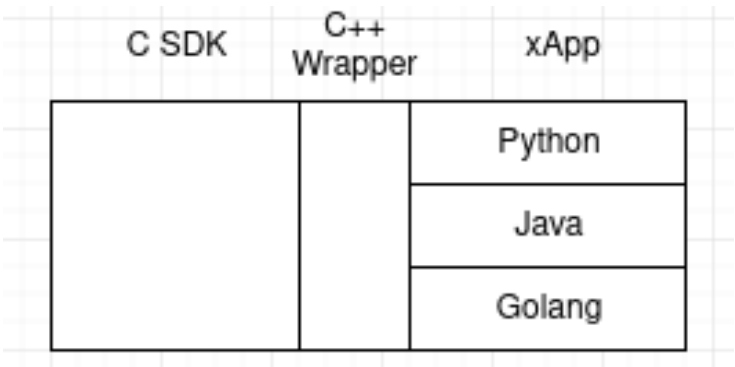



Figure: SWIG and the SDK

- For callbacks from C to Python, the Global Lock Interpreter (GIL) is needed.
- GIL is needed to avoid data races in python reference count memory model.
- This makes Python code single threaded when been called from the callback.
- Therefore, you probably don't want to spend much time in the callback

```
181
182 #ifdef XAPP_LANG_PYTHON
183     PyGILState_STATE gstate;
184     gstate = PyGILState_Ensure();
185 #endif
186
187     hndlr_rlc_cb->handle(&ind);
188
189 #ifdef XAPP_LANG_PYTHON
190     PyGILState_Release(gstate);
191 #endif
192
```

Figure: Python Global Lock Interpreter

# 5 Outline

# 5 FAQs I



- Why invent E42 instead of using the well tested gRPC?

# 5 Answer I

- gRPC (gRPC Remote Procedure Calls) is a cross-platform open source high performance Remote Procedure Call (RPC) framework from 2002.
- It uses HTTP/2 and Protobuffers which have been (mostly) overcomed by Flatbuffers.
- E42 uses a raw SCTP socket. The protocol is very similar to E2AP with slight modifications
- Currently, it encodes/decodes using ASN.1 but other possibilities are possible e.g., Flatbuffers

# 5 Answer I

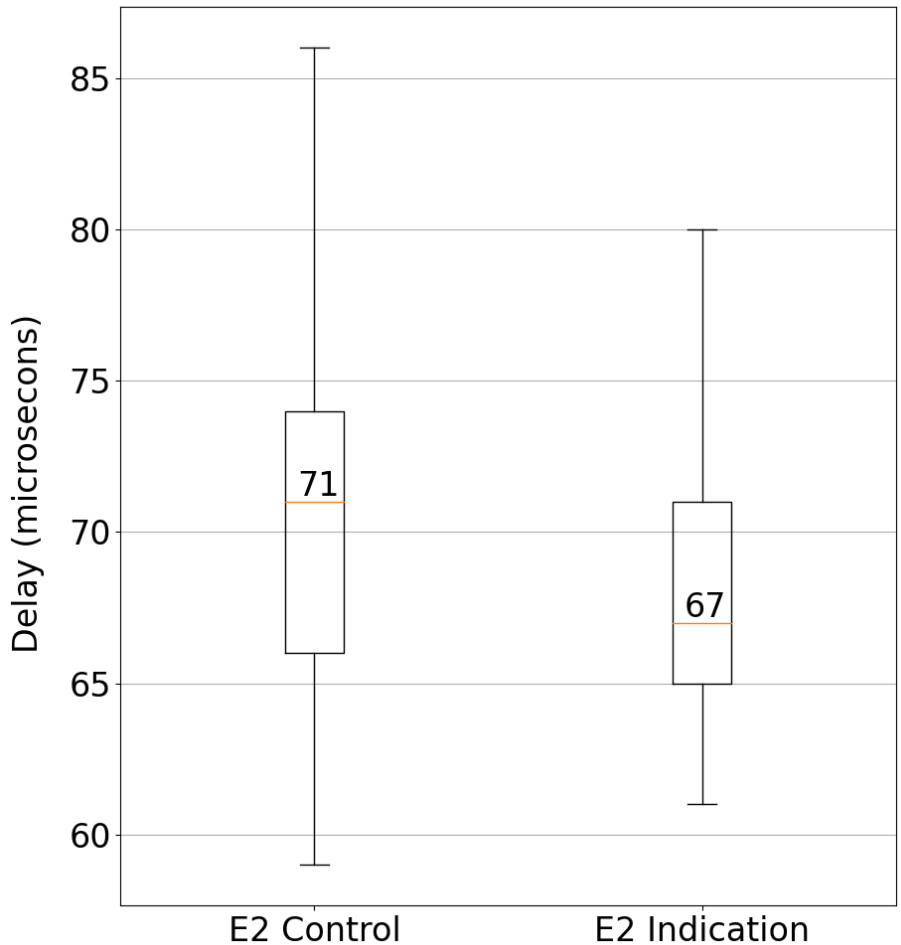



Figure: Latency of one xApp fetching data every 5 ms

# 5 FAQs II



- Why not use the O-RAN OSC developed E2AP? Why you developed a new E2AP version from scratch? Was it a waste of time?

# 5 Answer II



- ONOS, as well as O-RAN code was probably developed under time constraints sacrificing quality

```
304   RICaction_ToBeSetup_ItemIEs_t *action_item_ies = (RICaction_ToBeSetup_ItemIEs_t *)calloc(1, sizeof(RICaction_ToBeSetup_Item_t));
305   action_item_ies->id = proto_id;
306   action_item_ies->criticality = 0;
307
308   action_item_ies->value.present = pres6;
309   action_item_ies->value.choice.RICaction_ToBeSetup_Item.ricActionID = 5;
310   action_item_ies->value.choice.RICaction_ToBeSetup_Item.ricActionType = RICactionType_report;
311   action_item_ies->value.choice.RICaction_ToBeSetup_Item.ricActionDefinition = actdef;
312   action_item_ies->value.choice.RICaction_ToBeSetup_Item.ricSubsequentAction = sa;
```

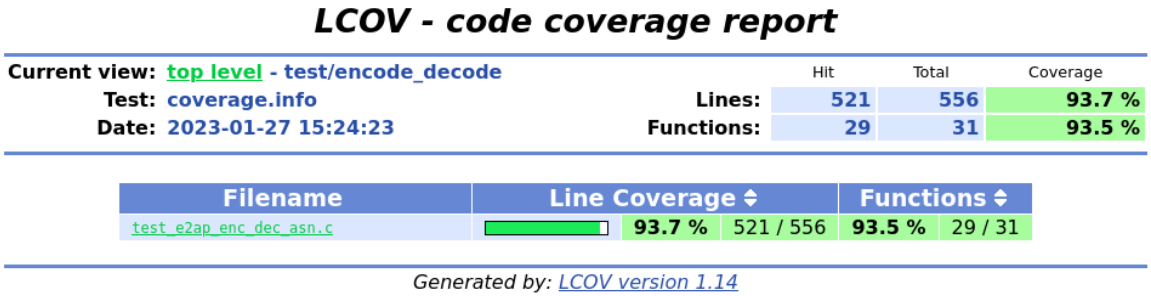
Figure: Code for Subscription Request from ONOS OAI repository fetched on the 27/01/203

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Message Type | M | | 9.2.3 | | YES | reject |
| RIC Request ID | M | | 9.2.7 | | YES | reject |
| RAN Function ID | M | | 9.2.8 | | YES | reject |
| RIC Subscription Details | M | | | | YES | reject |
| >RIC Event Trigger Definition | M | | 9.2.9 | | - | |
| >Sequence of Actions | | 1.. <maxofRICactionID> | | | EACH | ignore |
| >>RIC Action ID | M | | 9.2.10 | | - | |
| >>RIC Action Type | M | | 9.2.11 | | - | |
| >>RIC Action Definition | O | | 9.2.12 | | - | |
| >>RIC Subsequent Action | O | | 9.2.13 | | - | |

Figure: E2AP specification

# 5 Answer II

- FlexRIC is tested with a code coverage, a profiler, TSan and ASan.



**LCOV - code coverage report**

| | | Hit | Total | Coverage |
|---|---|---|---|---|
| Current view: top level - test/encode_decode | | | | |
| Test: coverage.info | Lines: | 521 | 556 | 93.7 % |
| Date: 2023-01-27 15:24:23 | Functions: | 29 | 31 | 93.5 % |

| Filename | Line Coverage | | Functions | |
|---|---|---|---|---|
| test_e2ap_enc_dec_asn.c | 93.7 % | 521 / 556 | 93.5 % | 29 / 31 |

Generated by: LCOV version 1.14

Figure: Code coverage of 20/26 E2AP functions on the 27/01/203

# 5 FAQs III



- Why FlexRIC was not developed as a cloud native software?

# 5 Answer III



- Definition: Cloud native computing is an approach in software development that utilizes cloud computing to "build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds"
- Zero overhead principle i.e., you don't pay for what you don't use i.e., no containers per default
- FlexRIC is easily deployable i.e., we do not use any 3rd party software
- FlexRIC can easily be decomposed (low-coupling, e.g., E42)

| Criteria | O-RAN RIC | ONOS-RIC | FlexRIC |
|---|---|---|---|
| **version** | OSC Bronze version | Jan 2022 version | Tag 0.1 |
| **Zero Overhead** | No | No | Yes |
| **Enc/Dec** | ANS1.c | ANS1.c | ANS1.c, FB, Plain |
| **E2AP version** | v2.0 | v2.0 | v1.0 |
| **E2AP IEs** | 7/26 | 10/26 | 20/26(ASN), 12/26( |
| **E2 Nodes** | Radisys | OAI, SRS, Radisys | OAI, SRS |
| **Memory footprint** | 1 | N/A | 0.17 |
| **E2Agent -xApp latency** | >1 ms | >1 ms | 0.1 ms |
| **Service Models** | Embedded | Embedded | Pluggable/Extenda |

Figure: Comparison between OSC, ONOS and FlexRIC

# 5 FlexRIC SDK



| Criteria | O-RAN SC RIC | ONF µONOS | FlexRIC xApp |
|---|---|---|---|
| version | 2022.07.31 | 2022.09.20 | Tag 1.0.0 |
| Near RT RIC to xApp SD interface | RMR/IS95 library | gRPC-base API | E42 API |
| xApp coupling with Near-RT RIC | Embedded | Embedded | Isolated |
| SM Coupling with Near-RT RIC | Tightly | Tightly | Loosely.<br>Plugins arch |
| SM and RF composability | No | No | Yes |
| Data Collection mechanism | Polling | Event Driven | Event Driven |
| xApp Baseline SDK | Go/Python/C++/Rust.<br>Language specific | Go/Python/C++/Rust.<br>Language specific | C<br>Language agnostic |
| xApp programming languages | 4 | 2 | 10+ |

Figure: Comparison between OSC, ONOS and FlexRIC

# 6 Outline

# 6 Live Coding



- Live coding!

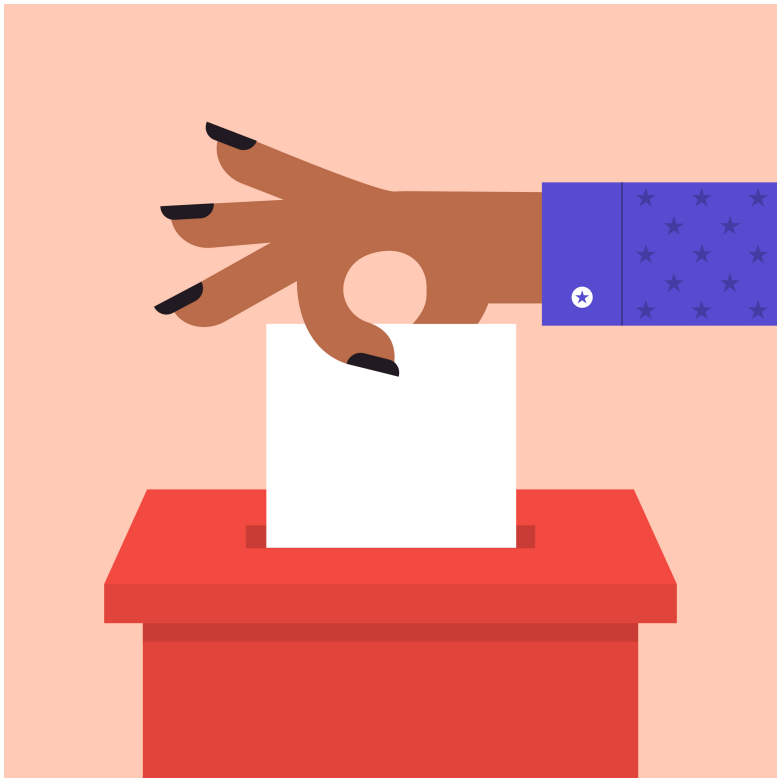
# 7 Outline

# 7 Q&A



# 7 FlexRIC Future

- Your feedback is needed!
- https://forms.gle/gFq1ycknjDw3qHRo8



- Subscribe to users@mosaic-5g.io or techs@mosaic-5g.io
- https://openairinterface.org/mosaic5g/

Thank you!