Workshop: Writing Python xApps for OSC's Platform

MSc. Student Daniel "Dante" Campos

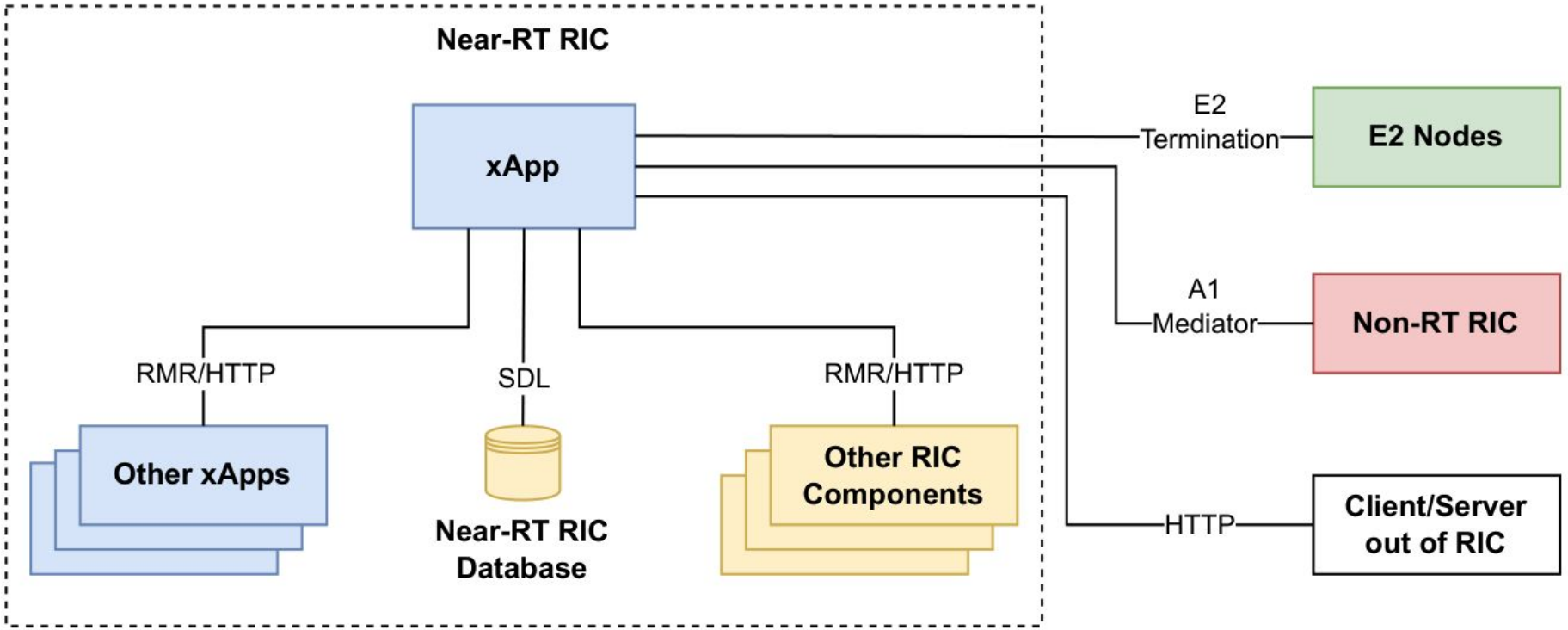
Computer Networks &
Distributed Systems
LABORAtory

INSTITUTO DE
INFORMÁTICA
UFG

Agenda

- Class 1: Managing, checking, and configuring xApps
- Class 2: xApp overview, logging, SDL, and REST
  - Common xApp endpoints
  - Writing xApps with OSC's ricxappframe package
  - xApp flow
  - Flow comparison: Xapp vs RMRXapp
  - Logging with OSC's mdclogpy package
  - SDL communication
  - REST communication
- Class 3: RMR communication and E2 Nodes subscription

Common xApp endpoints



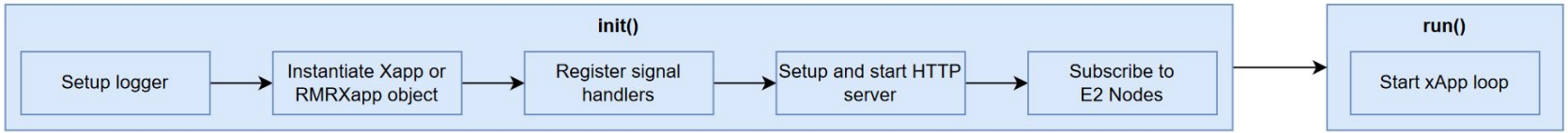
Writing xApps with OSC's ricxappframe package

- Defines **two xApp classes**
  - **Xapp**: general class for any type of xApp
  - **RMRXapp**: specific class for reactive xApps that act only when an event is detected, like changes in the config-file or RMR/HTTP messages received

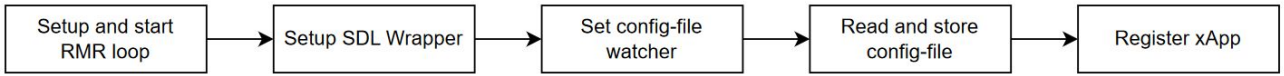
- Contains **functions** to:
  - Receive/send **RMR** messages
  - Interact with the **SDL**
  - Run HTTP servers for **REST** communication
  - Manage **E2 node** subscriptions

xApp flow

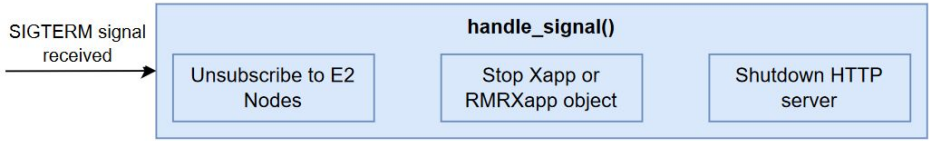
Custom xApp object flow



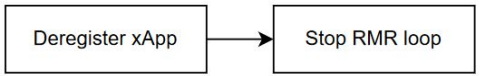
Xapp or RMRXapp object instantiation



Custom xApp object termination



Xapp or RMRXapp object stopping

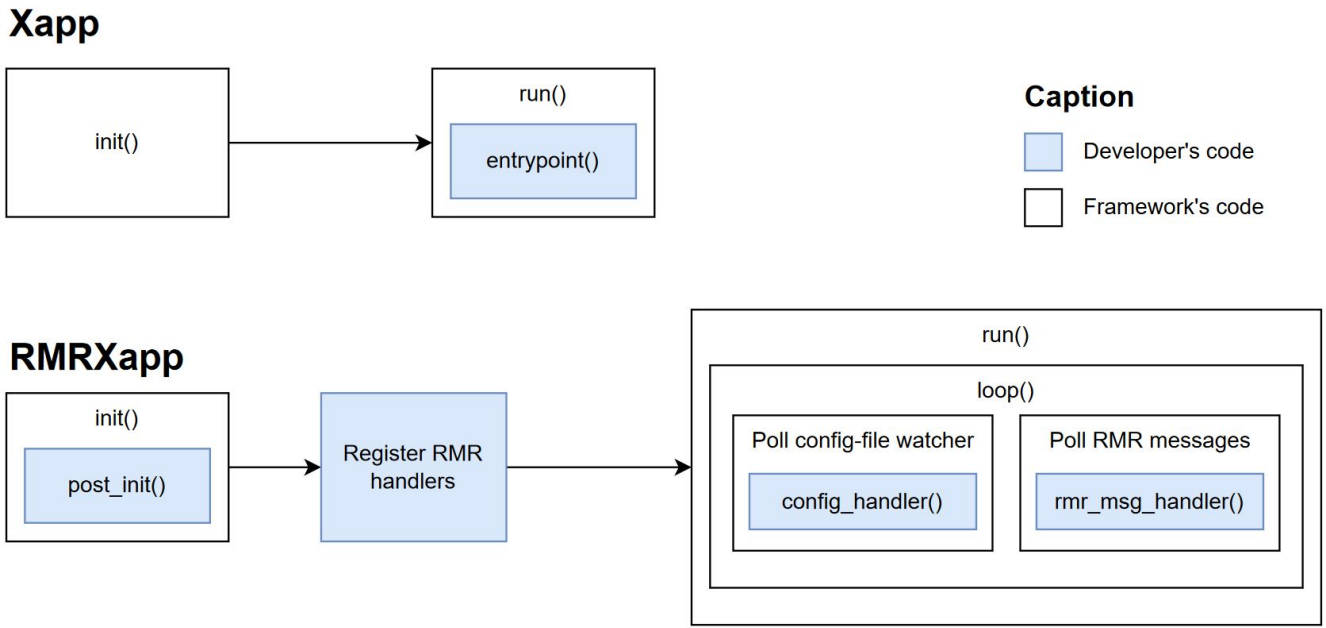


Caption

Developer's code

Framework's code

Flow comparison: Xapp vs RMRXapp



Logging with OSC's mdclogpy package

- Logs messages at **four levels of severity**
  - **Level hierarchy**: Error > Warning > Info > Debug
  - Only messages at the **actual level or above** are logged

- Supports **Mapped Diagnostic Context** (MDC)
  - A **dictionary** containing important xApp information for further debugging
  - The MDC can be **freely edited**

- Every log is a **JSON** with the fields:
  - **ts**: timestamp
  - **crit**: the log severity
  - **id**: logger name
  - **mdc**: the actual MDC
  - **msg**: message

SDL communication

- The SDL is an **interface** for interacting with the distributed RIC database
  - OSC's RIC cluster has a Database as a Service (DBaaS) pod running Redis DB
- The Redis DBaaS works like a **dictionary**
  - Every data is identified using a **key** and every key belongs to an **SDL namespace**
- Both Xapp and RMRXapp have access to the **SDL functions** below
  - **sdl_set**: stores data (overwrites)
  - **sdl_get**: retrives data
  - **sdl_find_and_get**: retrives all data whose keys start with a given prefix
  - **sdl_delete**: delete data

REST communication

- The xApp may use HTTP REST communication in **two ways**
  - **Sending** HTTP requests (calling functions from the **requests** package)
  - **Receiving** HTTP requests in an HTTP **server** (implemented with **ricxappframe.xapp_rest**)

- After setting up the HTTP server, we need to **register handlers**
  - Each handler will be called for a given **HTTP method** (GET, POST or DELETE) and **URI path**
  - The handler must **return a dictionary** with information (e.g. HTTP status) to send the **response**

REST communication

- **Main** xApp HTTP REST **requests to send**
  - **xApp registration**
    - Send POST to AppMgr’s HTTP service at path /ric/v1/xapps/register
    - Includes the xApp config-file
    - If the config-file is not sent, the AppMgr will request it to the xApp at the path /ric/v1/config
  - **xApp deregistration**
    - Send POST to AppMgr’s HTTP service at path /ric/v1/xapps/deregister

REST communication

- **Main** xApp HTTP REST **requests to receive**
  - **Config-file request**
    - Receive GET at path /ric/v1/config
    - Usually sent by AppMgr to get the xApp config-file if it is not in the registration request
  - **Liveness probe**
    - Receive GET at path /ric/v1/health/alive
    - Sent by Kubernetes periodically to check if the xApp is alive
    - The liveness probe path and periodicity must be specified in the config-file
  - **Readiness probe**
    - Receive GET at path /ric/v1/health/ready
    - Sent by Kubernetes periodically to check if the xApp is ready
    - The liveness probe path and periodicity must be specified in the config-file

References

OSC's xApp writer's guide v2: https://wiki.o-ran-sc.org/download/attachments/17269011/xApp_Writer_s_Guide_v2.pdf?version=4&modificationDate=1625642899082&api=v2

OSC's Python xApp Framework: https://pypi.org/project/ricxappframe/

OSC's mdclogpy Package: https://pypi.org/project/mdclogpy/

Polese, Michele, et al. "**Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges.**" IEEE Communications Surveys & Tutorials 25.2 (2023): 1376-1411.