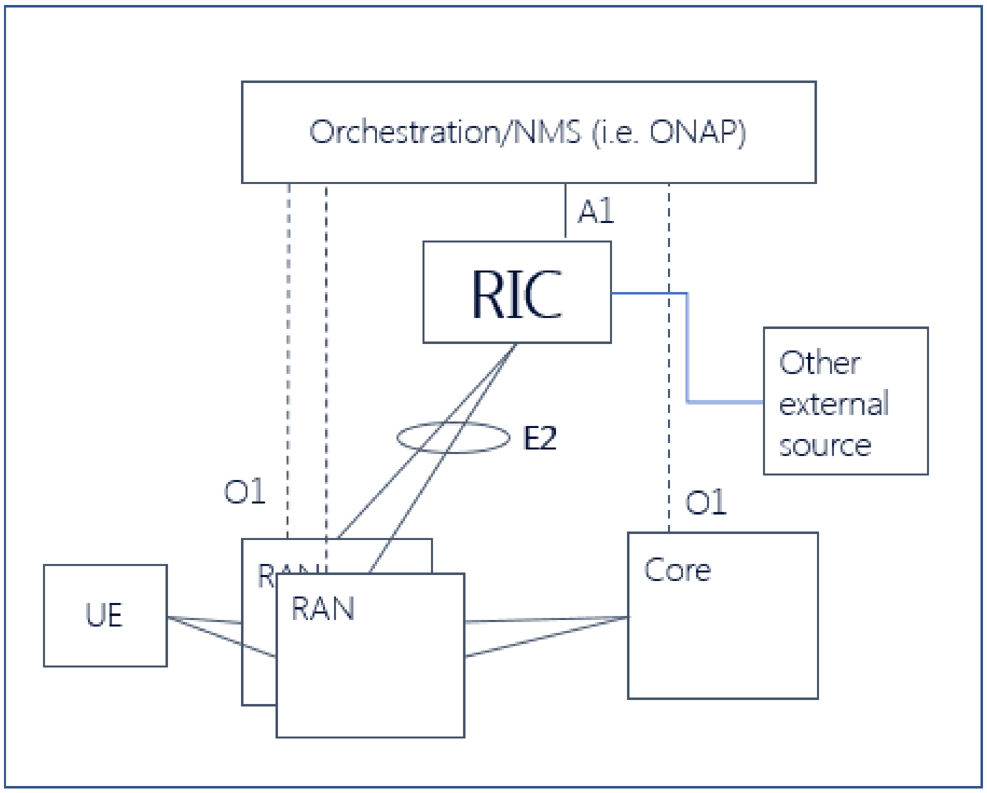
E2T Architecture

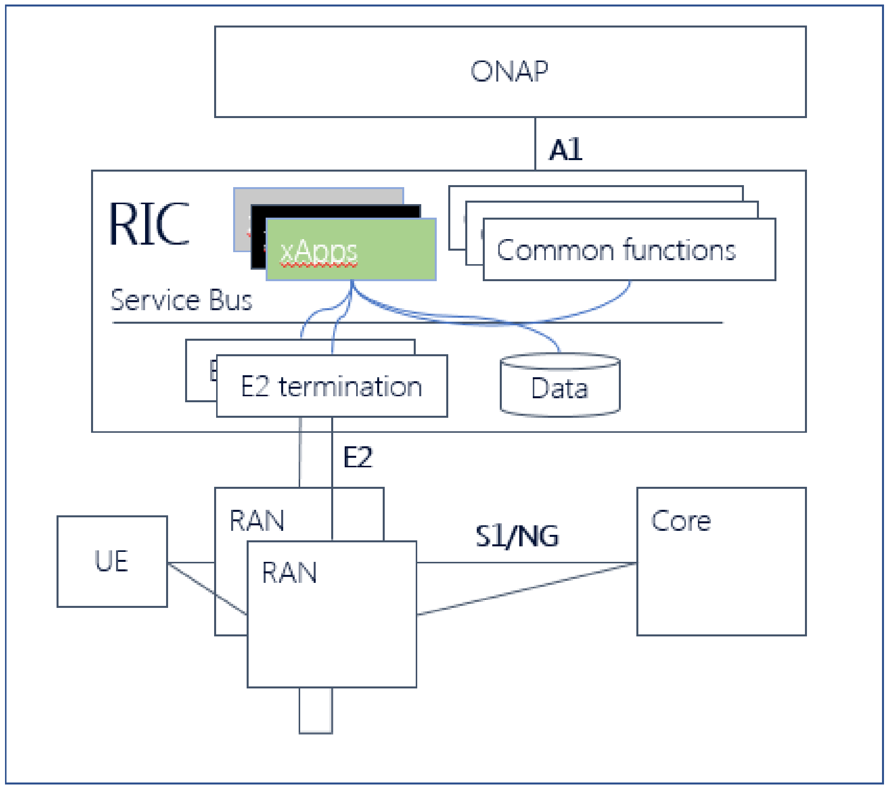
General

The purpose of this document is to describe the purpose of the E2 Terminator (E2T), how its architecture is built, what RAN methods it supports, what E2M / xAPP methods it supports.

RIC Hosts one or more “xApps” that use E2 interface to collect near Realtime information and provide value added services

RIC may obtain other information from “External sources” including ONAP, see Figure 1 RIC and its relation in the network.





RIC platform contains E2 termination, shared data and common functions to form the “RIC Platform”, running on a PaaS platform.

Common functions on RIC provide generic services (databases, ML tools, message bus, etc.) and have no role in RAN functionality

E2 termination is independent and binding messages to xApps, according RMR routing.

xApps, hosted on RIC Platform, provide RAN with services beyond functionality provided by BTS

RIC contains:

- E2 termination acting as RAN Function. The E2AP is done over SCTP Connection
- Central database assessible by xApps
- Multiple xApps including “E2 Manager” as xApp (0)

The service Bus, described in Figure 2 The RIC Platform, implemented by RMR.

Assumptions:

- RIC has E2 interface to one or more RAN nodes
- Each RAN node informs RIC of list of functions supporting RIC Services and the corresponding RIC Service Information Model
- xApps may address individual functions in specific RAN with messages routed via E2 termination in RIC and E2 agent in RAN
- Functions in specific RAN may respond to individual xApp in RIC with messages routed via E2 agent in RAN and E2 termination in RIC
- RIC E2 Termination acts as a RAN Function and provides both global procedures and RIC Services to a specialized “E2 Manager” xApp

ET Terminator Architecture

The E2T should supports the following:

- Open or remove SCTP Connection upon E2/EN-DC Setup or Removal
- Handle own repository to map RAN to SCTP Connection,
- Listen to all its SCTP Connection. Upon E2 method, decode it and sends it over the RMR to the relevant xAPP.
- Listen to the RMR Connection. Upon on RMR Request, convert it to E2AP, map the target RAN to SCTP Connection, decode it and sends it over the relevant SCTP.

Due to high performance and multiple connections, E2T is using epoll - a Linux kernel system call for a scalable I/O event notification mechanism. Its function is to monitor multiple file descriptors to see whether I/O is possible on any of them

So, E2T is one thread that listens (using epoll()) to any event coming from E2 or from RMR, and accordingly handles the message.