Interfacing DPL with raw FairMQDevices

Giulio Eulisse (CERN)

WP4 - 08th Dec 2017

Motivation

Not a fit for DPL: DPL is a layer on top of the `FairMQDevice` provided one and it's not meant to replace it, nor to completely abstract its functionalities. Even when the DPL is 100% complete, there will always be usecases which can only be solved by writing a FairMQDevice, by design.

Ease of transition: there is already a few devices which predate the DPL and some which even predate the O2 Data Model in the form currently specified by WP1. Being able to use those together with the DPL is a must to simplify transition from one to the other.

RawDeviceService

AliceO2Group/AliceO2#712 introduces a **RawDeviceService** which can be used by any computation to get hold of the underlying **FairMQDevice**. E.g.:

```
FairMQDevice *device = ctx.services().get<RawDeviceService>().device();
```

In order to create an out of band channel, this can now be specified as an extra **ConfigParamSpec** for a given **DataProcessorSpec**. E.g. to add channel named **foo** which connects to **localhost:8080** using Pub/Sub:

```
auto d = "name=foo,type=sub,method=connect,address=tcp://localhost:8080";
...
ConfigParamSpec{"channel-config", VariantType::String, d, "..."}
```

rawDeviceSource HELPER

What shown in the previous slide works well to extract data from the DPL, since this has to be tuned for the receiving device in any case.

For the case of injecting messages in the DPL a **rawDeviceSource** helper is provided, which creates a **DataProcessorSpec** and allows customization of the imported messages via a lambda.

```
auto outspec = OutputSpec{o2::Header::DataOrigin("SMPL"),
                          o2::Header::gDataDescriptionHeartbeatFrame};
WorkflowSpec workflow = {
  rawDeviceSource("foreign-source-name",
                  Outputs{outspec},
                  "type=sub,method=connect,address=tcp://localhost:5450",
                  o2DMAdaptor(outspec, 0, 1)
                 ),
  ...
```

Feedback responses

- I need to see the code to comment on the missing header / duplicate message issue.
- Debugging. Both gdb (well, lldb) and LOG should work fine. Again, we can sit together to understand the actual issue.
- This is indeed a bug. At the moment ports are statically allocated starting from 22000. Clean solution would be to integrate the driver with DDS (as Dennis showed in the offline week). I can think of a quick workaround if this is a showstopper.
- Forwarding data: you should be able to do this by getting the `DataRef` from the inputs and using `DataAllocator::adoptChunk`. We can make this first class API in `DataAllocator`.