# ALFA: Status and Plans

**Mohammad Al-Turany**

**GSI scientific computing**

# Simulation:
# Arbitrary Types with FairRoot/IO

# Arbitrary Types with FairRoot/IO

## Status quo

- So far, only classes of type TNamed or TCollections could be exchanged with the FairRootManager for persistent IO and exchange of data between tasks

Registering outgoing data

```
TClonesArray *mHits;
mHits = new TClonesArray("tpc::Hit");
iomgr->Register("TPCHit", mHits, true);
```

```
mHits = dynamic_cast<TClonesArray*>(iomgr->GetObject("TPCHit"));
```

Asking for incoming data

- Memory overhead due to
  - Data elements needs to be TObject (~16bytes overhead)
  - Data elements are stored as pointers (8bytes overhead)
- Type unsafe code
  - TClonesArray can't do type checking at compile time

# Arbitrary Types with FairRoot/IO

## New feature

- A pull request has been made to FairRoot which extends the FairRootManager to handle branches of any type for the purpose of IO and data exchange

Store hits in stl containers

Registering typed outgoing data

```
std::vector<tpc::Hit> *mHits;
iomgr->RegisterAs("TPCHit", mHits, true);
```

```
mHits = iomgr->InitObjectAs<std::vector<tpc::Hit>*>("TPCHit");
```

Asking for typed incoming data

- No negative effect on split level of persistent branches
- Feedback/comments on PR welcome
- Once PR accepted, I would like to move completely away from TClonesArrays in favour of std::vectors

A Large Ion Collider Experiment

## Arbitrary Types with FairRoot/IO

### New feature

- A pull request has been m… FairRootManager t… of IO and da…

Ability to register/retrieve data of any type with IO manager

#636 MohammadAlTu… merged 3 commits into FairRootGroup:dev from sawenzel:swenzel/new on Oct 5

Merged

Store hits in stl containers

Registering typed outgoing data

```
…s;
…hit", mHits, true);
```

```
…->InitObjectAs<std::vector<tpc::Hit>*>("TPCHit");
```

Asking for typed incoming data

See talk by Sandro in Combined WP12/WP13 meeting, 04/10/17
https://indico.cern.ch/event/670362/

- Once PR accepted, I would like to move completely away from TClonesArrays in favour of std::vectors

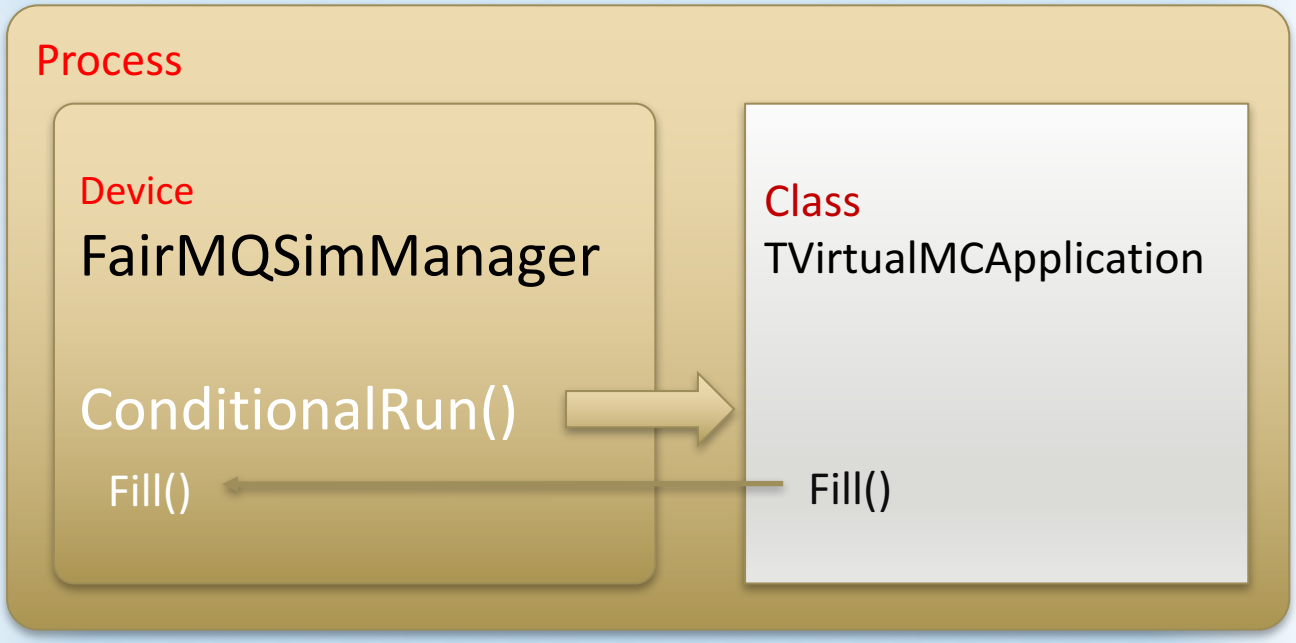
# Simulations in FairMQ

Radoslaw Karabowicz

- Start Simulation as set of FairMQ device:
  - Manage the parameter via the FairParMQServer(): Use the same parameter interface in simulation and reconstruction
  - Digitizers/User processes could also run in separate device

# Proof of concept: Implementation

Radoslaw Karabowicz




PROTOTYPE

# ParameterMQServer

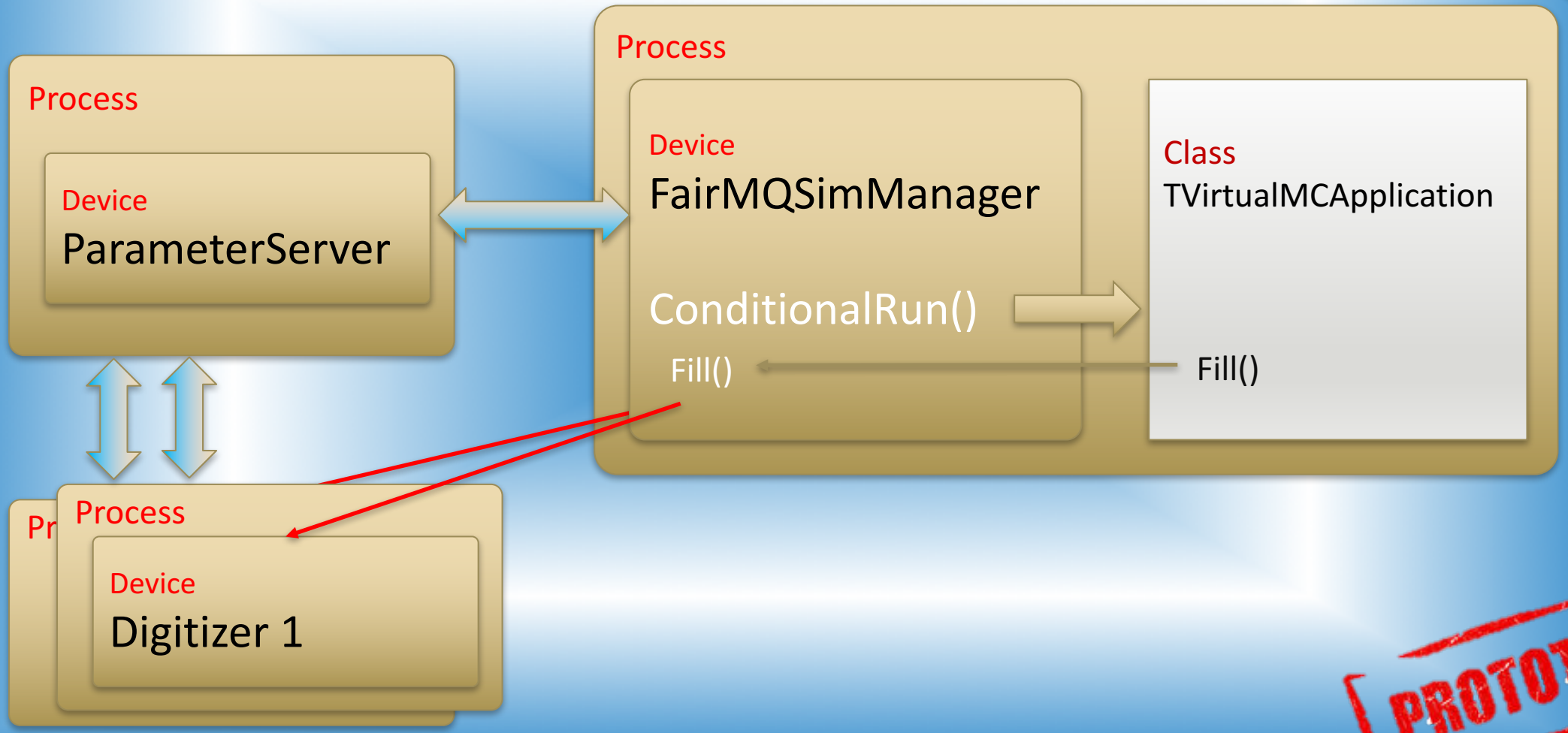
Radoslaw Karabowicz

OnData() for two channels is implemented:

- ProcessRequest()
  - sends back the requested parameter in reply to the request of any device connected to this channel (previous behavior of the ParameterServer);
- ProcessUpdate()
  - accepts parameters sent on this channel (plus special actions after receiving TObjString: run id changes, save request).

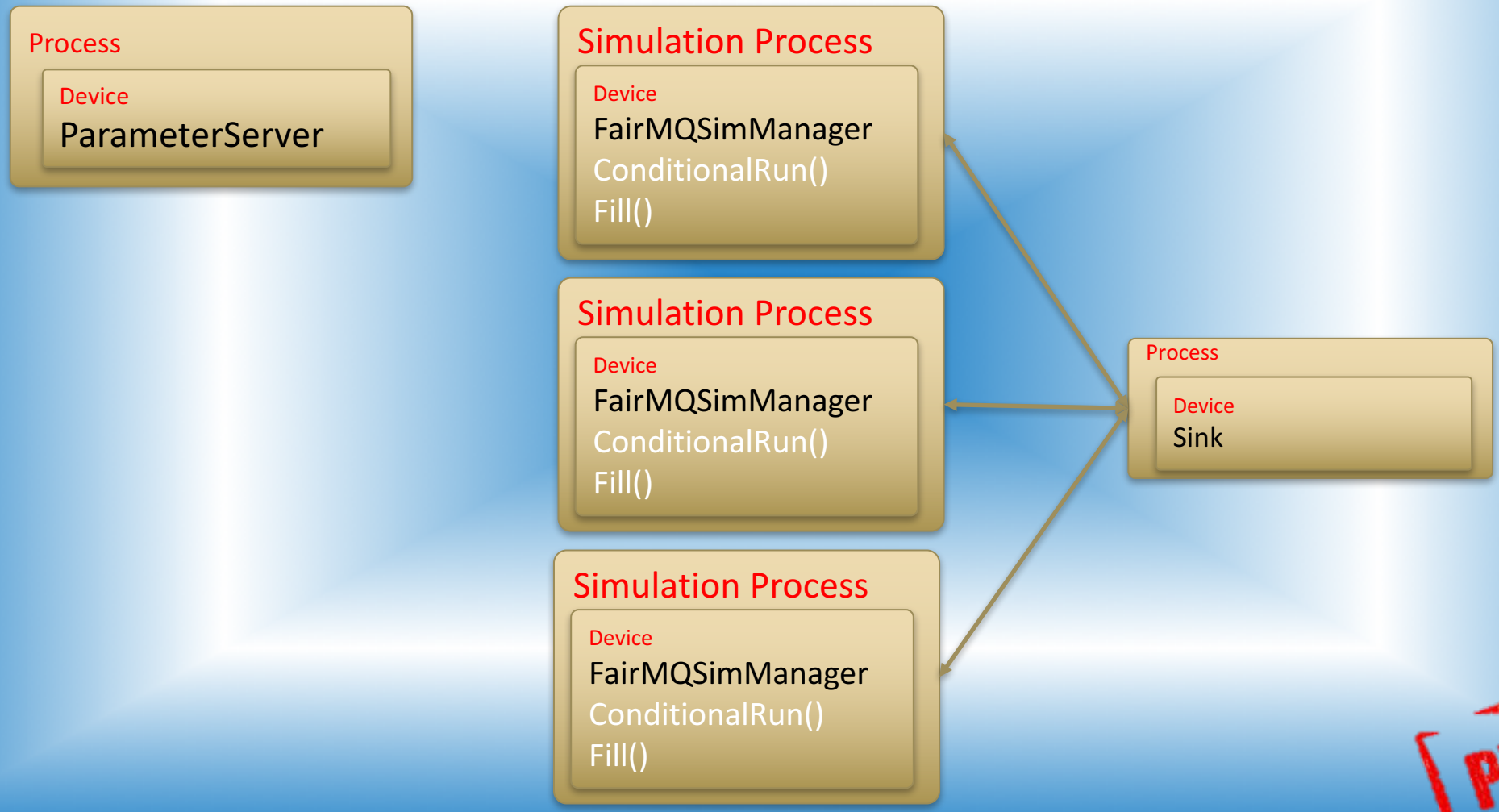
# Putting the things together

Radoslaw Karabowicz



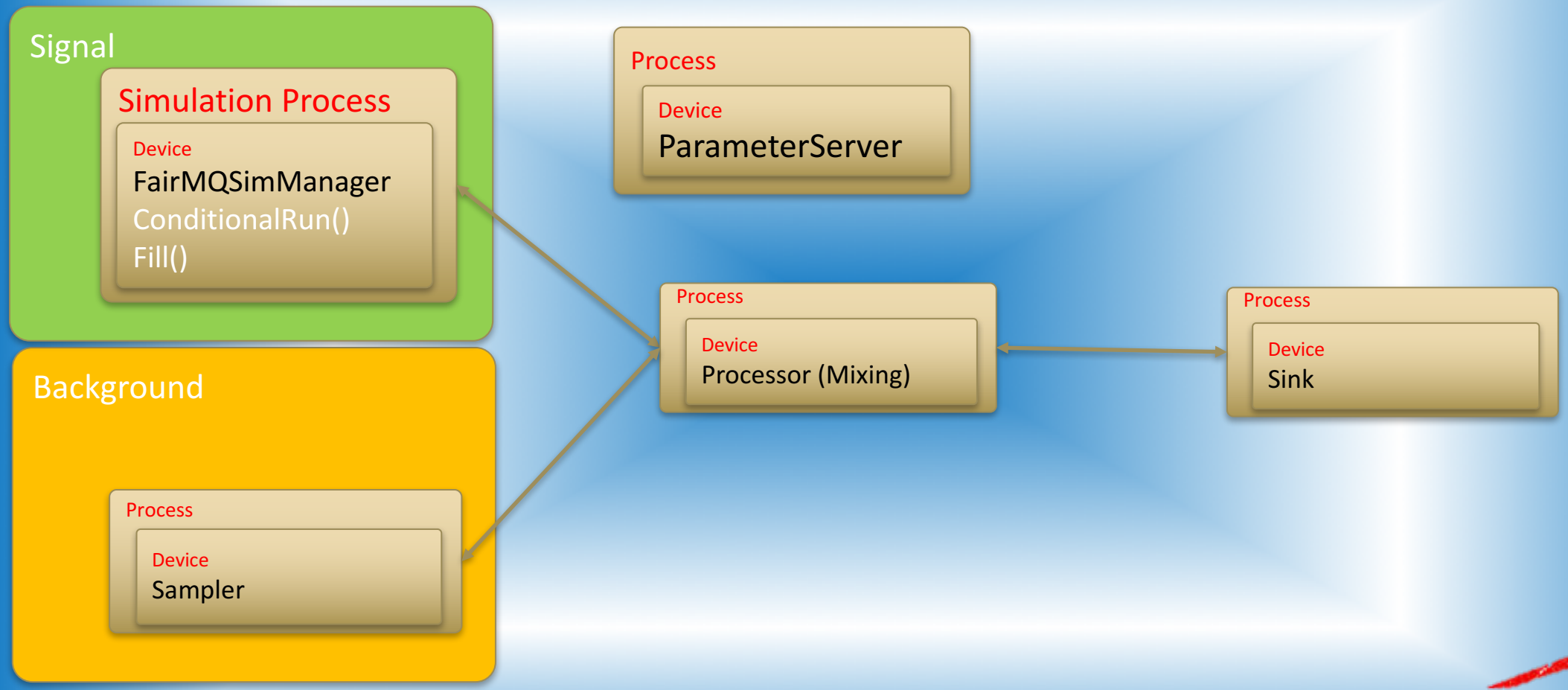

PROTOTYPE

# When the simulation is too fast!



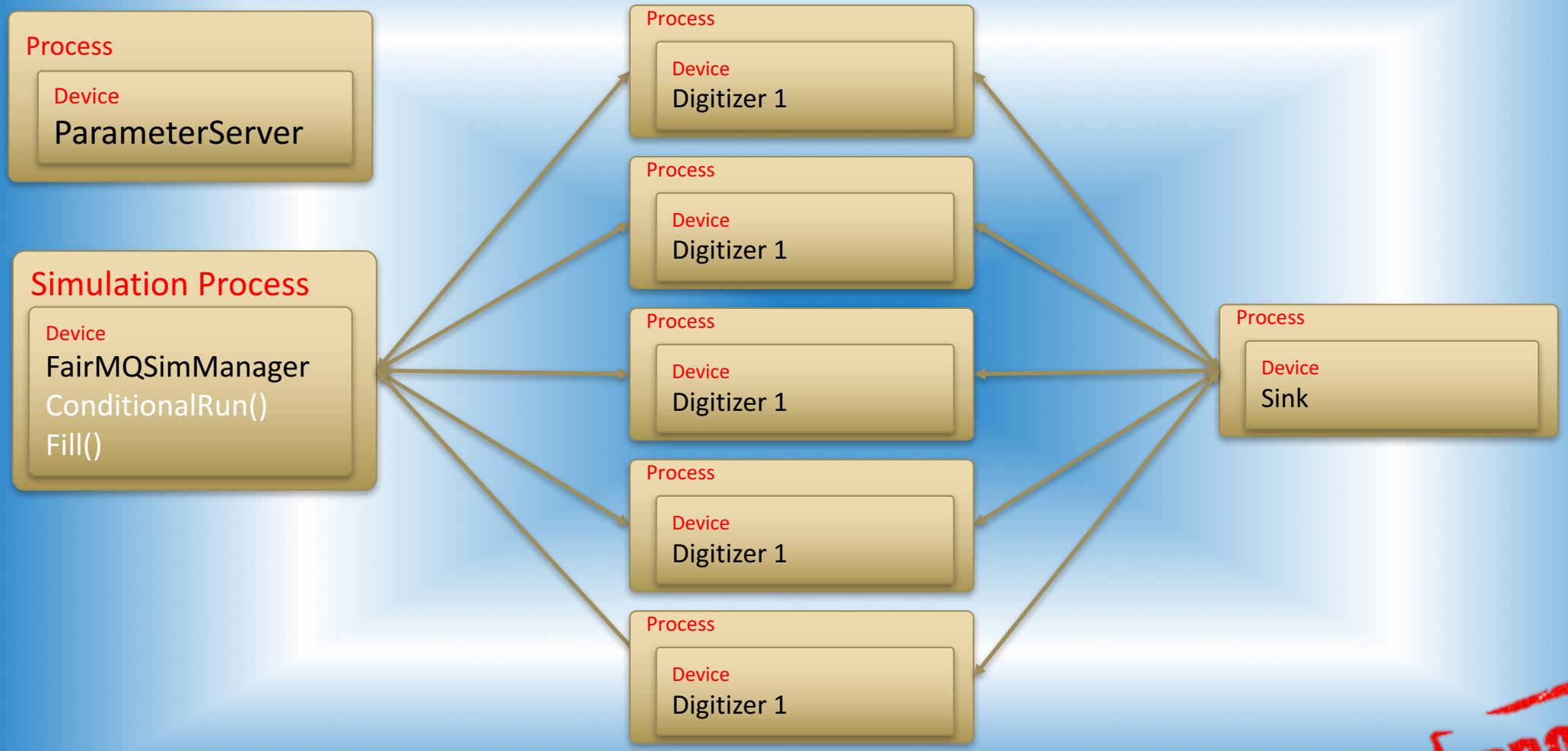

PROTOTYPE

# Could also help in mixing!




PROTOTYPE

# Parallization of simulation and digitization




PROTOTYPE

# Simulation in FairMQ:
# Different topologies for simulations can be adapted to the problem itself and the existing resources

# FairMQ

# What is new?

- Unmanaged shared memory region
- Plugin ins :  see talk by Dennis
- DDS:  see talk by Andrey

# FairMQ: General concepts

- Hide all transport-specific details from the user.
- Clean, maintainable and extendable interface to different data transports (ZMQ, nanomsg, shared memory ...).
- Allow combinations of different transport in one device in a transparent way.
- Any device/channel can switch transport only via configuration, without modifying device/user code -> same API.

# FairMQ: Ownership concepts

- Message owns data.
- Sender device (user code) passes ownership of data to framework with send call.
- Framework transfers to next device, passes ownership to receiver.
- No sharing of ownership between different devices.

# Unmanaged Shared Memory Region

# Unmanaged Shared Memory Region:

- Give user full freedom with the memory layout of the data (which can be required by certain hardware, e.g CRU driver, CRI (CBM), ..
- But leave the allocation of the region to the transport, to ensure most efficient transfer to further steps

# How it works:

- User can request memory region from a transport.
- Place data structures in the region, and
- Create messages out of subsets of the region.

https://github.com/FairRootGroup/FairRoot/tree/master/examples/advanced/Region

# Messages from a Region

- Region messages are not cleaned up automatically (thus the name unmanaged),
- Only the entire region is cleaned up once the corresponding object is destructed.
- Cleanup of the buffers of region messages is left to the user entity that manages the region contents.
- Receiving devices see messages originating from a region just as a normal message - the region details are hidden behind the scenes.

# In development

- A callback mechanism to notify the user when a region message buffer is no longer needed by the transport(s) of all relevant devices (e.g.: use by shared memory and/or RDMA) – similar to APIs that already exist for user provided buffers.

# What is next?

- Integration of the full chain in AliceO2 (more news in the Plenary next week)
- RDMA Transport
- Re-design of the whole CMake infrastructure in ALFA-FairRoot