New API for serialisation in ALFA

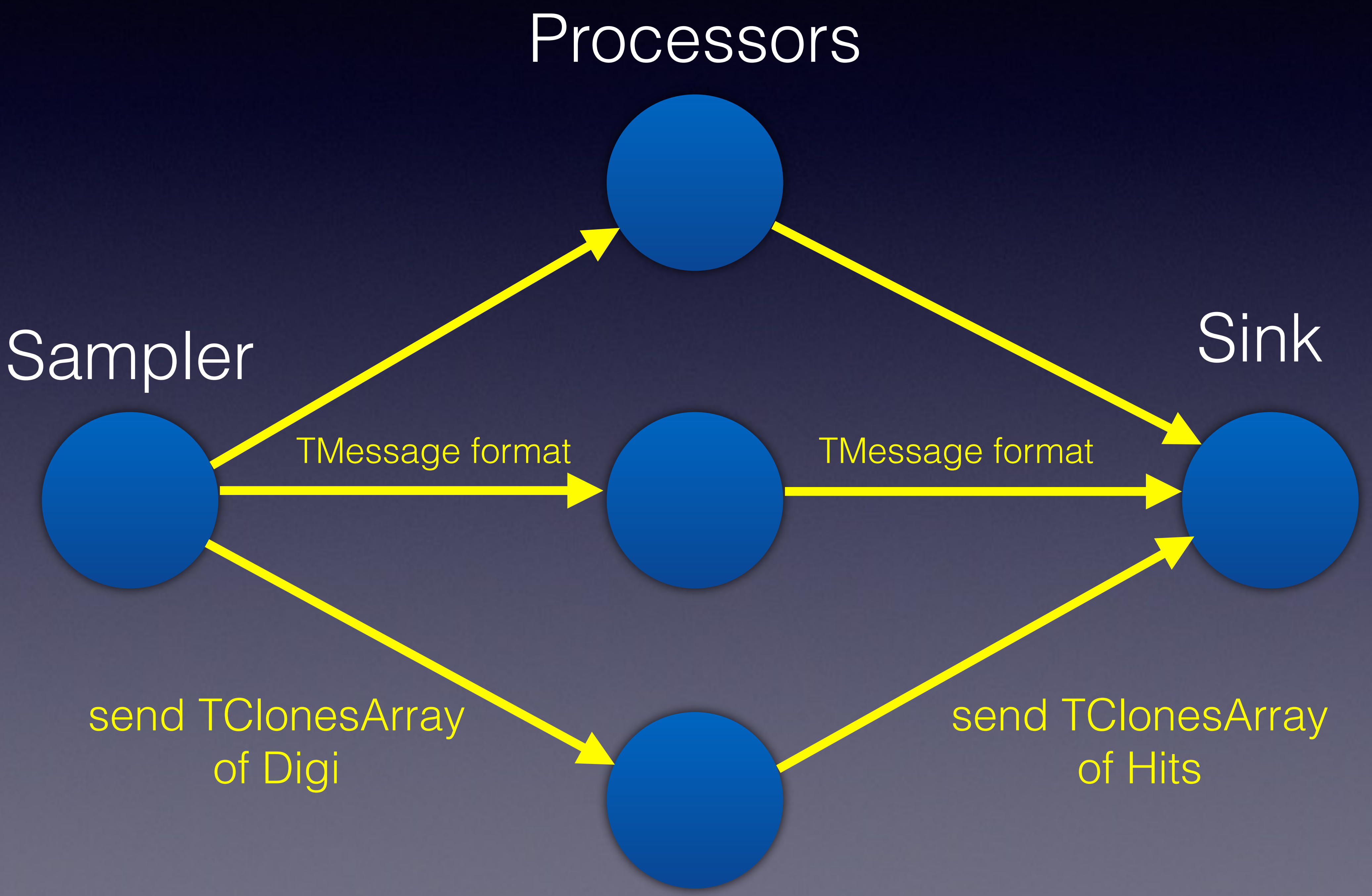
Nicolas Winckler
GSI-Darmstadt

ALICE Offline Week - 1/4/2016

- FairMQ does not support serialization/deserialization
- Instead an abstract serialization interface has been introduced
- Some requirements:
  - Interface should be flexible and accept any data type, and eventually, any additional argument
  - Interface should be able to call any external serialization methods
  - Interface should work with multi-part API

Example 1 (single-message case)

See: FairRoot/examples/MQ/serialization/src/1-simple



Call example in a Processor (single-message case)

```
// ...
virtual void Run()
{
    while (CheckCurrentState(RUNNING))
    {
        std::unique_ptr<FairMQMessage> msg(NewMessage());
        if (Receive(msg,"data-in") > 0)
        {
            Deserialize<MyDeserializer>(*msg,fInput);
            Exec(fInput,fOutput);
            Serialize<MySerializer>(*msg,fOutput);
            Send(msg, "data-out");
        }
    }
}
// ...
private:
TClonesArray* fInput;
TClonesArray* fOutput;
```

See : FairRoot/examples/MQ/serialization

Serializer example using Root-TMessage

```
struct MySerializer
{
    void Serialize(FairMQMessage& msg, TClonesArray* input)
    {
        TMessage* tm = new TMessage(kMESS_OBJECT);
        tm->WriteObject(input);
        msg->Rebuild(tm->Buffer(), tm->BufferSize(), free_tmessage, tm);
    }
};

struct MyDeserializer
{
    void Deserialize(FairMQMessage& msg, TClonesArray*& output)
    {
        if(output) delete output;
        FairTMessage tm(msg.GetData(), msg.GetSize());
        output = static_cast<TClonesArray*>(tm.ReadObject(tm.GetClass()));
    }
};
```

FairMQDevice::Serialize method implementation

```
template<typename Serializer, /* explicit template parameter */
          typename DataType, /* deduced template parameter */
          typename... Args      /* deduced variadic template parameter(s) */
         >
inline void Serialize(FairMQMessage& msg, DataType&& data, Args&&... args) const
{
   Serializer().Serialize(msg, std::forward<DataType>(data), std::forward<Args>(args)...);
}
```

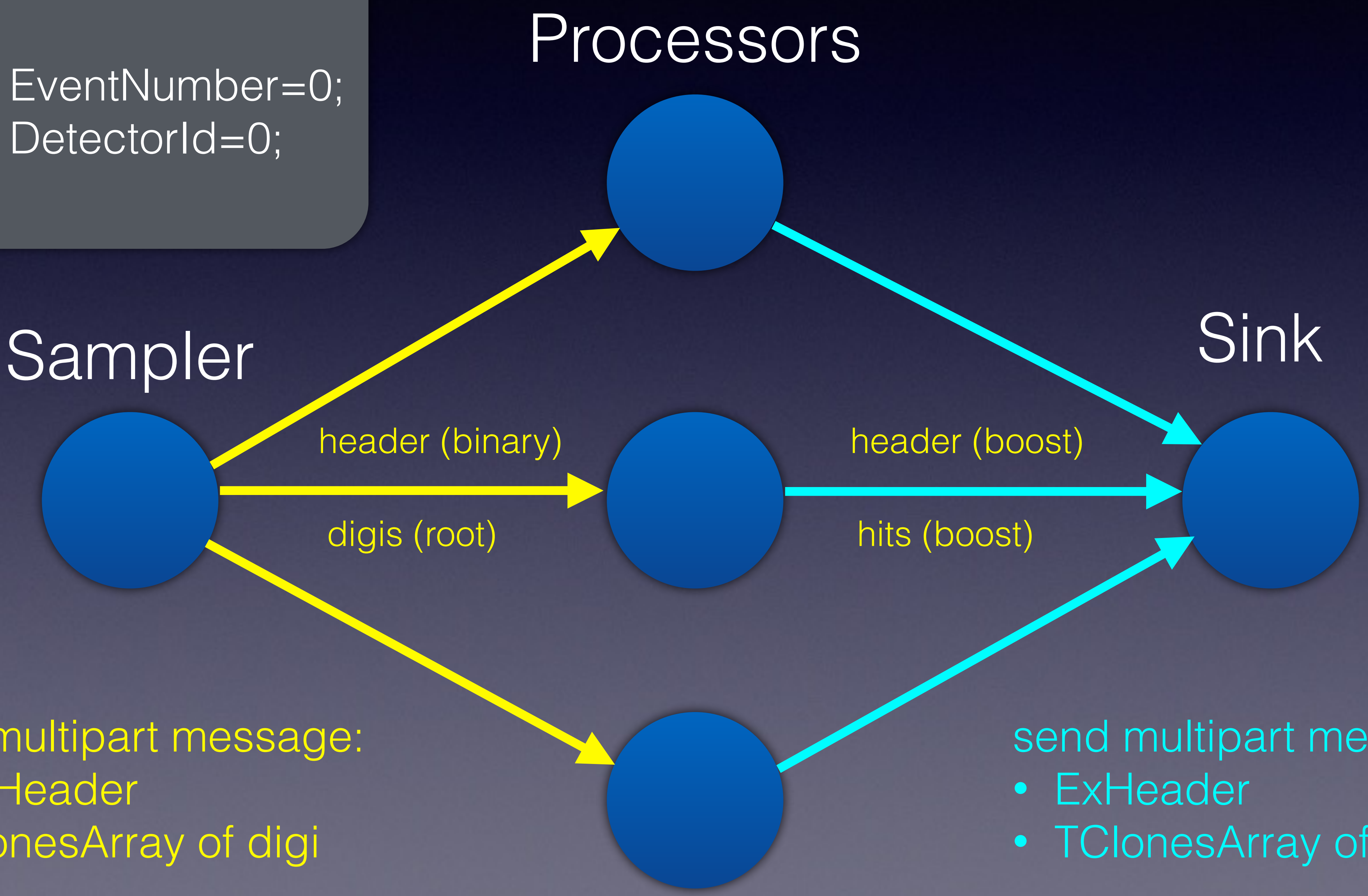
```
template<typename Deserializer,
          typename DataType,
          typename... Args
         >
inline void Deserialize(FairMQMessage& msg, DataType&& data, Args&&... args) const
{
   Deserializer().Deserialize(msg, std::forward<DataType>(data), std::forward<Args>(args)...);
}
```

- Interface allows to build flexible interface at compile time e.g.
  - **OneSerializer**::**Serialize**(FairMQMessage& msg, DataType1* data)
  - **AnotherSerializer**::**Serialize**(FairMQMessage& msg, DataType2*& data, Arg1Type* arg1, const Arg2Type& arg2, Arg3Type arg3)
- One can also use
  - the same explicit template argument for both, FairMQDevice::Serialize/Deserialize methods
    - **MySerializer::Serialize**(FairMQMessage& msg, const DataType& data)
    - **MySerializer::Deserializer**(FairMQMessage& msg, DataType& data)
  - overloads of SerializerType::Serialize/Deserialize methods
    - **MySerializer::Deserializer**(FairMQMessage& msg, DataType& data)
    - **MySerializer::Deserializer**(FairMQMessage& msg, HeaderType& header)

Example 2 (multi-part-message case)

In FairRoot/examples/MQ/serialization/src/2-multi-part

```
struct Ex2Header
{
    int EventNumber=0;
    int DetectorId=0;
};
```



send multipart message:
- Ex2Header
- TClonesArray of digi

send multipart message:
- ExHeader
- TClonesArray of Hits

Call example (multi-part-message case)

```
// ...
virtual void Run()
{
    while (CheckCurrentState(RUNNING))
    {
        FairMQParts parts;
        if (Receive(parts,"data-in") > 0)
        {
           Ex2Header* header=nullptr;
           Deserialize<SerializerEx2>(parts.At(0),header);
           Deserialize<SerializerEx2>(parts.At(1),fInput);
           Exec(fInput,fOutput);

           FairMQParts partsToSend;
           Serialize<SerializerEx2Boost>(partsToSend.At(0),*header);
           Serialize<SerializerEx2Boost>(partsToSend.At(1),fOutput);
           Send(partsToSend, "data-out");
        }
    }
}
// ...
private:
TClonesArray* fInput;
TClonesArray* fOutput
```

See: FairRoot/examples/MQ/serialization/src/2-multi-part

Serializer example using Boost

```
struct SerializerEx2Boost{

    void Serialize(FairMQMessage& msg, const Ex2Header& header)
    {
        std::ostringstream buffer;
        BoostBinArchOut OutputArchive(buffer);
        OutputArchive << header;
        std::string* strMsg = new std::string(buffer.str());
        msg->Rebuild(const_cast<char*>(strMsg->c_str()), strMsg->length(), my_deleter, strMsg);
    }

    void Deserialize(FairMQMessage& msg, Ex2Header& header)
    {
        std::string msgStr(static_cast<char*>(msg->GetData()), msg->GetSize());
        std::istringstream buffer(msgStr);
        BoostBinArchIn InputArchive(buffer);
        InputArchive >> header;
    }
};
```

Summary

- Users can easily create their own adaptors to plugin their serialization into the FairMQ/ALFA workflow.

Deserializer example

```
struct MyDeserializer{
   void Deserialize(FairMQMessage& msg, DataType& data)
              {/*deserialization code*/}
};
struct MySerializer{
   void Serialize(FairMQMessage& msg, const DataType& data)
              {/*serialization code*/}
};
struct MyOtherSerializer{
   void Serialize(FairMQMessage& msg, DataType* data, ArgType arg)
              {/*serialization code*/}
   void Deserialize(FairMQMessage& msg, DataType*& data)
              {/*deserialization code*/}
};
```

Run example

```
if (Receive(msg,"data-in") > 0)
{
   Deserialize<MyDeserializer>(msg,fInput);
   Exec(fInput,fOutput);
   Serialize<MySerializer>(msg,fOutput)
   Send(msg, "data-out");
}
```

- **Boost and ROOT serializer** available in FairRoot/base/MQ/policy/serialization

- Examples in FairRoot/examples/MQ/serialization