

FairMQ Status & Plugin demonstrators

Dennis Klein

Scientific IT
GSI Darmstadt

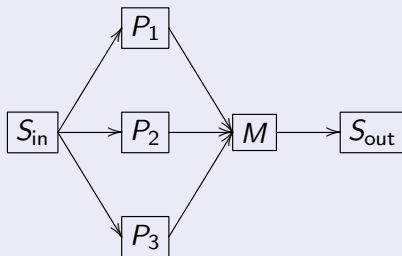
Alice Offline Week 10th November 2017
CERN

Outline

- 1 Recap
- 2 New APIs
- 3 Demonstrators
- 4 Status Update
- 5 Outlook

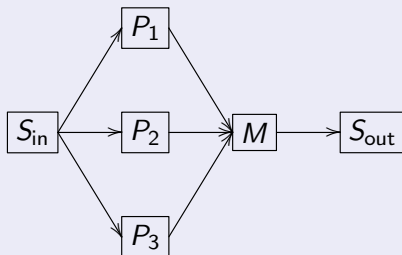
Recap

FairMQ topology



Recap

FairMQ topology



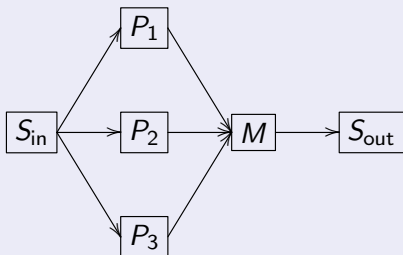
needs to be integrated with

External services

Control Config Monitoring ...

Recap

FairMQ topology



⇒

needs to be integrated with

External services

Control Config Monitoring ...

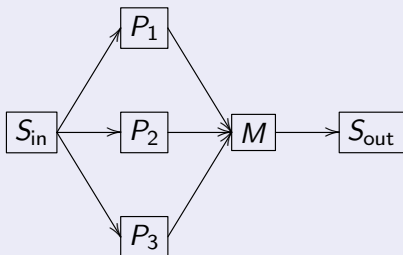
Add **client functionality** to external services to FairMQ devices.

Requirements

- Optional
- Dynamic dependencies
- Development in own repo

Recap

FairMQ topology



⇒

needs to be integrated with

External services

Control Config Monitoring ...

Add **client functionality** to external services to FairMQ devices.

Requirements

- Optional
- Dynamic dependencies
- Development in own repo

⇒ calls for classic plugin mechanism

Already implemented and released in Q3 2017

- This talk is about follow-ups and further improvements since then.

Event-based config plugin API I

Templated value type:

```
template<typename T>
void SubscribeToPropertyChange(
    std::function<void(const std::string& key, T newValue)> callback);

template<typename T>
void UnsubscribeFromPropertyChange();
```

⇒ Seperate subscription per value type needed.

Event-based config plugin API II

Auto conversion to strings:

```
void SubscribeToPropertyChangeAsString(  
    std::function<void(const std::string& key,  
                        std::string  newValue)> callback);  
void UnsubscribeFromPropertyChangeAsString();
```

Event-based config plugin API II

Auto conversion to strings:

```
void SubscribeToPropertyChangeAsString(  
    std::function<void(const std::string& key,  
                        std::string newValue)> callback);  
void UnsubscribeFromPropertyChangeAsString();
```

Requirement

Custom property types must provide an overload for operator<< in the form

```
friend std::ostream& operator<< (std::ostream& stream,  
                                const MyType& value);
```

(Analog to std::string GetPropertyAsString(const std::string& key)
const)

config key discovery plugin API

```
std::vector<std::string> GetPropertyKeys() const;
```

Returns the list of all property keys currently known by the device.

config key discovery plugin API

```
std::vector<std::string> GetPropertyKeys() const;
```

Returns the list of all property keys currently known by the device.

Usage example (in combination with synchronous config plugin API)

```
// Discover device config
for (const auto& key: GetPropertyKeys()) {
    LOG(DEBUG) << key << GetProperty(key);
}
```

Take/ReleaseDeviceControl plugin API

Only one plugin should be able to control (the state machine of) the device at a time.

```
void TakeDeviceControl();  
void StealDeviceControl();  
void ReleaseDeviceControl();
```

Calls to the API above in combination with the plugin load/instantiation order determines, which plugin becomes “Device Controller”.

Take/ReleaseDeviceControl plugin API

Only one plugin should be able to control (the state machine of) the device at a time.

```
void TakeDeviceControl();  
void StealDeviceControl();  
void ReleaseDeviceControl();
```

Calls to the API above in combination with the plugin load/instantiation order determines, which plugin becomes “Device Controller”.

Usage example (in plugin constructor)

```
try {  
    TakeDeviceControl();  
}  
catch (PluginServices::DeviceControlError& e) {  
    // If we are here, it means another plugin has taken control.  
}
```

```
#include <runFairMQDevice.h>
```

Reminder

<runFairMQDevice.h> ...

- ... allows to write device executables with minimal boilerplate, and
- ... offers two customization points (CLI options, device creation),

```
#include <runFairMQDevice.h>
```

Reminder

<runFairMQDevice.h> ...

- ... allows to write device executables with minimal boilerplate, and
 - ... offers two customization points (CLI options, device creation),
-
- \Rightarrow but is not flexible enough for advanced use cases, e.g.
 - O2 workflow executor implements much more complex executables and spawns devices deeper down the call stack.

```
#include <fairmq/DeviceRunner.h> |
```

```
fair::mq::DeviceRunner runner{argc, argv};
```

```
runner.AddHook<LoadPlugins>([] (DeviceRunner& r){  
    r.fPluginManager->SetSearchPaths({"lib", "lib/plugins"});  
    r.fPluginManager->LoadPlugin("asdf"); });
```

```
runner.AddHook<SetCustomCmdLineOptions>([] (DeviceRunner& r){  
    boost::program_options::options_description customOptions("CustomOptions");  
    // ...  
    r.fConfig.AddToCmdLineOptions(customOptions); });
```

```
runner.AddHook<ModifyRawCmdLineArgs>([] (DeviceRunner& r){  
    r.fRawCmdLineArgs.push_back("--blubb"); });
```

```
runner.AddHook<InstantiateDevice>([] (DeviceRunner& r){  
    r.fDevice = std::make_shared<MyFairMQDevice>(); });
```

```
return runner.RunWithExceptionHandlers(); // or just runner.Run()
```

```
#include <fairmq/DeviceRunner.h> |
```

```
fair::mq::DeviceRunner runner{argc, argv};

runner.AddHook<LoadPlugins>([] (DeviceRunner& r){
    r.fPluginManager->SetSearchPaths({"lib", "lib/plugins"});
    r.fPluginManager->LoadPlugin("asdf"); });

runner.AddHook<SetCustomCmdLineOptions>([] (DeviceRunner& r){
    boost::program_options::options_description customOptions("Custom options");
    // ...
    r.fConfig.AddToCmdLineOptions(customOptions); });

runner.AddHook<ModifyRawCmdLineArgs>([] (DeviceRunner& r){
    r.fRawCmdLineArgs.push_back("--blubb"); });

runner.AddHook<InstantiateDevice>([] (DeviceRunner& r){
    r.fDevice = std::make_shared<MyFairMQDevice>(); });

return runner.RunWithExceptionHandlers(); // or just runner.Run()
```

```
#include <fairmq/DeviceRunner.h> |
```

```
fair::mq::DeviceRunner runner{argc, argv};

runner.AddHook<LoadPlugins>([] (DeviceRunner& r){
    r.fPluginManager->SetSearchPaths({"lib", "lib/plugins"});
    r.fPluginManager->LoadPlugin("asdf"); });

runner.AddHook<SetCustomCmdLineOptions>([] (DeviceRunner& r){
    boost::program_options::options_description customOptions("CustomOptions");
    // ...
    r.fConfig.AddToCmdLineOptions(customOptions); });

runner.AddHook<ModifyRawCmdLineArgs>([] (DeviceRunner& r){
    r.fRawCmdLineArgs.push_back("--blubb"); });

runner.AddHook<InstantiateDevice>([] (DeviceRunner& r){
    r.fDevice = std::make_shared<MyFairMQDevice>(); });

return runner.RunWithExceptionHandlers(); // or just runner.Run()
```

```
#include <fairmq/DeviceRunner.h> //
```

- DeviceRunner currently has four customization points (calls your function at certain points in time during device startup/shutdown sequence)
- DeviceRunner allows integration of devices into more complex executable implementations with fairly few boilerplate
- <runFairMQDevice.h> is now implemented with a DeviceRunner

```
#include <fairmq/DeviceRunner.h> ||
```

- DeviceRunner currently has four customization points (calls your function at certain points in time during device startup/shutdown sequence)
- DeviceRunner allows integration of devices into more complex executable implementations with fairly few boilerplate
- <runFairMQDevice.h> is now implemented with a DeviceRunner

Takeaway

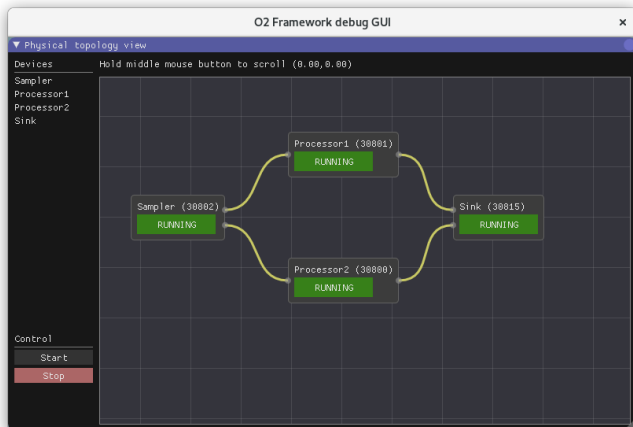
- ① Keep using `#include <runFairMQDevice.h>` by default.
- ② Use DeviceRunner if you need more flexibility.
- ③ Approach us, if you have a use case you cannot solve with DeviceRunner

examples/MQ/3-dds

- Demo
- <https://github.com/FairRootGroup/FairRoot/tree/dev/examples/MQ/3-dds>
written by Alexey over 1.5 years ago and maintained during that time
- only added [o] dump config to fairmq-dds-command-ui for today's demo

O2 DDS workflow executor + DebugGUI

• Demo



Status Update

- Event-based config, key discovery, device control plugin APIs **done**
- DeviceRunner **done**
- Porting DDS plugin to new system **done**
- Moving builtin device controllers (static, interactive) to builtin plugin (statically linked and always loaded) **done**

Status Update

- Event-based config, key discovery, device control plugin APIs **done**
- DeviceRunner **done**
- Porting DDS plugin to new system **done**
- Moving builtin device controllers (static, interactive) to builtin plugin (statically linked and always loaded) **done**
- Alexey's progress covered by Mohammad's talk

Status Update

- Event-based config, key discovery, device control plugin APIs **done**
- DeviceRunner **done**
- Porting DDS plugin to new system **done**
- Moving builtin device controllers (static, interactive) to builtin plugin (statically linked and always loaded) **done**
- Alexey's progress covered by Mohammad's talk

See

- https://github.com/FairRootGroup/FairMQPlugin_example for a FairMQ plugin skeleton,
- <https://github.com/FairRootGroup/FairRoot/tree/dev/fairmq/plugins> for more complex plugin examples,
- <https://github.com/FairRootGroup/FairRoot/tree/dev/fairmq/README.md> for general FairMQ docs.

Outlook

- RDMA transport **in progress** (Q4 2017 planned, more likely Q1 2018)

Outlook

- RDMA transport **in progress** (Q4 2017 planned, more likely Q1 2018)
- Major FairRoot build system modernization **in progress** (Q4 2017 at dev)
 - FairRoot CMake package
 - Full imported/exported targets support
 - Better AliBuild integration
 - ROOT dictionary generation from targets
 - and more, details at <https://github.com/FairRootGroup/FairRoot/pulls/632>

Outlook

- RDMA transport **in progress** (Q4 2017 planned, more likely Q1 2018)
- Major FairRoot build system modernization **in progress** (Q4 2017 at dev)
 - FairRoot CMake package
 - Full imported/exported targets support
 - Better AliBuild integration
 - ROOT dictionary generation from targets
 - and more, details at <https://github.com/FairRootGroup/FairRoot/pulls/632>
- Plugin log API **in progress** (Q4 2017)
- Plugin monitoring API **open**

Outlook

- RDMA transport **in progress** (Q4 2017 planned, more likely Q1 2018)
- Major FairRoot build system modernization **in progress** (Q4 2017 at dev)
 - FairRoot CMake package
 - Full imported/exported targets support
 - Better AliBuild integration
 - ROOT dictionary generation from targets
 - and more, details at <https://github.com/FairRootGroup/FairRoot/pulls/632>
- Plugin log API **in progress** (Q4 2017)
- Plugin monitoring API **open**

Thank you for your attention! Any questions?