

# FairMQ

## *Status & Plans*

Alexey Rybalchenko  
(GSI Darmstadt, FairRoot group)

***ALICE Offline Week***  
***CERN, December 7, 2018***

# FairMQ

Developments since last Offline Week

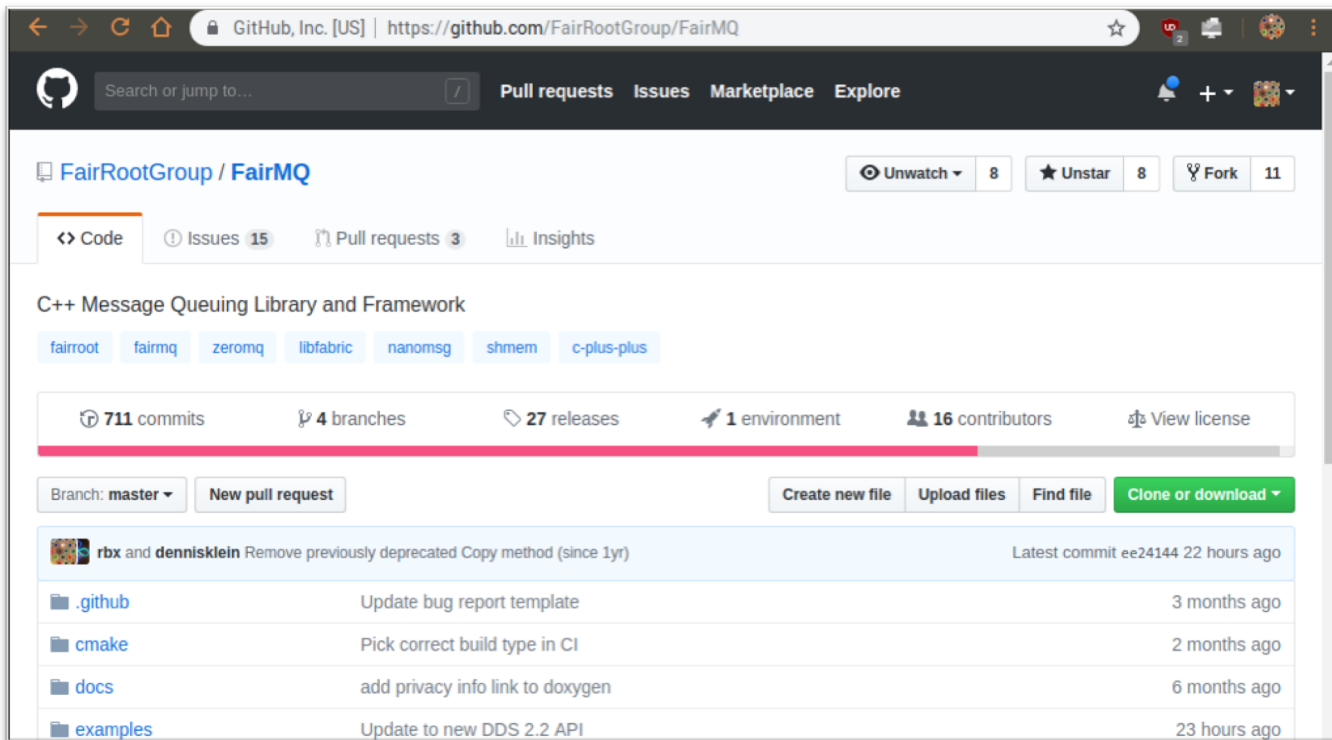
- **FairMQ repository**
- **CMake modernization**
- **Transports status**
- **New features/enhancements**
- **Deprecations**
- **Tests**
- **Future plans**

# FairMQ

## own repository

Allow easier use of FairMQ without need for FairRoot and its dependencies.  
More flexibility in the development/release cycle.

- FairLogger also moved to its own repository, because it is a dependency of both FairRoot and FairMQ.
  - Stable FairRoot release without FairMQ – v18.0.0



FairMQ

<https://github.com/FairRootGroup/FairMQ/>

current stable release: **v1.3.7**  
released on 29-11-2018

**v1.3.7 release notes**

<https://github.com/FairRootGroup/FairMQ/releases/tag/v1.3.7>

# FairMQ

## CMake modernization

FairMQ now fully embraces modern CMake idioms.  
Same improvements are coming to FairRoot and AliceO2.

Bonus feature: better CMake output

More details:

CMake WP3 talk by Dennis Klein

<https://indico.cern.ch/event/684244/>

Previous Offline Week, talk by Dennis Klein

<https://indico.cern.ch/event/710009/>

FairMQ-specific usage details:

<https://github.com/FairRootGroup/FairMQ/blob/master/README.md>

Patches submitted to ease CMake integration

<https://github.com/VcDevel/Vc/commits?author=dennisklein&since=2018-08-31&until=2018-09-30>

<https://github.com/msgpack/msgpack-c/commits?author=dennisklein&since=2018-07-31&until=2018-08-31>

<https://github.com/nanomsg/nanomsg/commits?author=dennisklein&since=2018-08-31&until=2018-09-30>

<https://github.com/zeromq/libzmq/commits?author=dennisklein&since=2018-04-25&until=2018-08-31>

<https://github.com/FairRootGroup/DDS/commits?author=dennisklein&since=2018-08-31&until=2018-09-30>

```
~/d/FairMQ > dev build
> cmake -DCMAKE_INSTALL_PREFIX="/Users/orybalch/dev/FairMQ/install/" -DCMAKE_PREFIX_PATH="/Users/orybalch/dev/install/FairSoft_dev/" -DBUILD_DDS_PLUGIN=ON -DBUILD_NANOMSG_TRANSPORT=ON ..
-- FairMQ 1.3.7 from Wed Nov 28 21:16:29 2018 +0100
--
-- BUILD TYPE      CXX FLAGS
-- Debug           -g -Wshadow -Wall -Wextra
-- Release         -O2 -DNDEBUG
-- * RelWithDebInfo -O2 -g -Wshadow -Wall -Wextra -DNDEBUG
-- Nightly         -O2 -g -Wshadow -Wall -Wextra
-- Profile         -g3 -Wshadow -Wall -Wextra -fno-inline -fctest-coverage -fprofile-arcs
-- Experimental   -O2 -g -Wshadow -Wall -Wextra -DNDEBUG
-- AddressSan      -O2 -g -Wshadow -Wall -Wextra -fsanitize=address -fno-omit-frame-pointer
-- ThreadSan      -O2 -g -Wshadow -Wall -Wextra -fsanitize=thread
--
-- (Change the build type with -DCMAKE_BUILD_TYPE=...)
--
-- DEPENDENCY FOUND  VERSION      PREFIX
-- Boost             1.67 (>= 1.64)  /Users/orybalch/dev/install/FairSoft_dev
-- FairLogger        1.3.0 (>= 1.2.0) /Users/orybalch/dev/install/FairSoft_dev
-- ZeroMQ            4.2.5 (>= 4.1.5) /Users/orybalch/dev/install/FairSoft_dev
-- nanomsg           1.1.5 (>= 1.1.3) /Users/orybalch/dev/install/FairSoft_dev
-- msgpack           3.1.0 (>= 3.1.0) /Users/orybalch/dev/install/FairSoft_dev
-- DDS               2.2 (>= 2.2)   /Users/orybalch/dev/install/FairSoft_dev
-- GTest             unknown (>= 1.7.0) /Users/orybalch/dev/install/FairSoft_dev
--
-- COMPONENT  BUILT?  INFO
-- fairmq     YES    (default, disable with -DBUILD_FAIRMQ=OFF)
-- tests      YES    (default, disable with -DBUILD_TESTING=OFF)
-- nanomsg_transport YES    (disable with -DBUILD_NANOMSG_TRANSPORT=OFF)
-- ofi_transport NO    EXPERIMENTAL (default, enable with -DBUILD_OFI_TRANSPORT=ON)
-- dds_plugin  YES    (disable with -DBUILD_DDS_PLUGIN=OFF)
-- examples   YES    (default, disable with -DBUILD_EXAMPLES=OFF)
-- docs       NO    (default, enable with -DBUILD_DOCS=ON)
--
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/orybalch/dev/FairMQ/build
~/d/FairMQ > dev build
```

1004ms < Thu Nov 29 15:33:42 2018

# FairMQ

## Transports status

- **ZeroMQ** (default transport).
- **Shared Memory**: introduced in FairRoot v17.03, **ready to use**, more feedback welcome.
- **OFI** → following talk by Dennis Klein! (Offline Week 2018-12-07, 09:30)
- **nanomsg** (currently maintained mainly for comparison & testing).

The development of nanomsg itself is in bugfix-only mode, in favor of a reimplementaion known as nng (nanomsg-next-generation). Currently no further investigation of nng ongoing.

node	process	address format	ZeroMQ <sup>[5]</sup>	nanomsg <sup>[6]</sup>	shmem	OFI <sup>[4]</sup>
intra-	intra-	inproc://endpoint	✓	✓	n/a	n/a
intra-	inter-	ipc://endpoint	✓	✓	✓	X
		tcp://host:port	✓	✓	✓	✓
inter-	inter-	tcp://host:port	✓	✓	n/a	✓
		verbs://host:port	X	X	n/a	✓

✓ - zero-copy    ✓ - not zero-copy    X - no support planned

CHEP  
Posters

**CHEP-2018 Poster #366**

RDMA-accelerated data transport in ALFA

<https://indico.cern.ch/event/587955/contributions/2938030/>

**CHEP-2018 Poster #305**

Shared Memory Transport for ALFA

<https://indico.cern.ch/event/587955/contributions/2938141/>

# Avoid data copy when switching transports

(if a transport supports this)

After the introduction of **transport mixing in FairMQ** (e.g. ZeroMQ and shared memory in the same device), when a message created by one transport was given to another it was copied by the framework.

This is actually necessary in many cases because some transports require to allocate its buffers itself.

However, some transports are also **able to accept a foreign buffer without a copy**.

**FairMQ now allows a transport to avoid this copy if it is capable of doing so.** This is done in a general way without any transport needed to know others. It only needs to know if it can accept a user buffer.

Current most common case: a **shared memory** message that is given to **ZeroMQ** transport – ZeroMQ can accept user buffer without an additional copy. In this case the shared memory will be released once ZeroMQ is finished with the message buffer.

Upcoming use case: an RDMA message that can be written directly to shared memory (could require both transports to know each other).

# Run state handlers on the main thread

## feature

Request from WP12

- Until recently FairMQ executed all user state handlers (InitTask, Run, etc..) in a dedicated thread, different from the main thread.
- However, certain user code (in particular Geant4-MT) requires code to be executed on the same thread.
- To support this, **FairMQ now executes user state handlers on the main thread.**
- The device controller (typically a control plugin that initiates and responds to state changes) has to run in a different thread instead, to keep the device responsive to control.

**This is a non-breaking change for anyone who uses our provided `main()/control plugin` or a custom `main()` with DeviceRunner (most users).**

For anyone else:

- Call **`device.RunStateMachine()`** on the main thread (or any other thread you want it to run on).
- Control it from a separate thread via the unchanged control API.

Example: <https://github.com/FairRootGroup/FairMQ/pull/66/files#diff-72b0af547968f1404b947b700d60af83>

# DDS command UI

## target parts of the topology

DDS talk by Andrey Lebedev  
(Offline Week 2018-12-07, 10:00)

FairMQ provides a control/configuration plugin based on DDS.

A command utility is provided that allows user to get devices configuration, check or change device states.

FairMQ v1.3.7 extends this utility to be able to send commands not only to all devices at once, but to target parts of the running topology using the DDS topology path.

### Example:

### Topology:

```
...  
<main id="main">  
  <task>Sampler</task>  
  <task>Sink</task>  
  <group id="ProcessorGroup" n="10">  
    <task>Processor</task>  
  </group>  
</main>  
...
```

### Command UI:

#### send command to sampler:

```
$ fairmq-dds-command-ui -s <session-id> -c c -p main/Sampler
```

#### send command to all processors:

```
$ fairmq-dds-command-ui -s <session-id> -c c -p main/ProcessorGroup/Processor
```

#### send command to processor #4:

```
$ fairmq-dds-command-ui -s <session-id> -c c -p main/ProcessorGroup/Processor_4
```

etc...



# New Rate Limiter

Requirement: be able to efficiently limit the rate of producer devices: `--rate` cmd option (value in Hz).`

## Previously:

(1) Our benchmark sampler device (fairmq-bsampler) accepts `--msg-rate` argument to limit the sending rate. It implements this in an efficient way, but requires an additional thread to be run.`

(2) Similar feature submitted as a general device argument (`--rate` by Matthias Richter for any device that uses ConditionalRun(). It works well and is single-threaded, but has performance disadvantages with very high rates (from 200kHz upwards) due to time measurement overhead.`

## Now:

A new implementation submitted by Matthias Kretz that does not require an additional thread and performs very well for high rates (by avoiding time measurements in many iterations). This implementation is now used in both cases.

### Implementation

<https://github.com/FairRootGroup/FairMQ/blob/master/fairmq/tools/RateLimit.h>

# Interactive Logging Control

logger/control plugin feature

The interactive device controller now supports real-time log severity & log verbosity switching via key inputs:

[k] increase log severity, [l] decrease log severity, [n] increase log verbosity, [m] decrease log verbosity

Corresponding APIs (both FairLogger and Plugins have the same API):

```
static void CycleConsoleSeverityUp();  
static void CycleConsoleSeverityDown();  
static void CycleVerbosityUp();  
static void CycleVerbosityDown();
```

# Deprecate Send-/ReceiveAsync

## Use timeout overload instead

In the effort of simplifying our APIs we are deprecating the `SendAsync` and `ReceiveAsync` methods. Their names are misleading of what these methods actually do. Both methods try to perform a Send/Receive operation on the underlying transport, if the current queue condition allows it, or signal the caller if this is not possible (queue full/empty). A more fitting name would be `TrySend/TryReceive`.

However, same can also be achieved via slightly changed mechanics of the existing timeout value of the regular Send/Receive API, with a timeout value of 0.

```
int FairMQChannel::Send(FairMQMessagePtr& msg, int sndTimeoutInMs = -1);
```

The `sndTimeoutInMs` value is either:

- 1: will try to queue the message until success or until interrupted.
- 0: will try to queue the message once and signal caller immediately if the queue is full.
- >0: will try to queue the message for <value> milliseconds.

Same mechanics for Receive, but instead of trying to fill the queue it will try to get a message from the queue.

**Example:** `SendAsync(msg)` becomes `Send(msg, 0)`.

# Tests Coverage

- Improving test coverage to cover all of our public APIs and internal code (coverage from v1.2.x -> v1.3.7: ~60% -> ~80%).
- Unit test suites & functional tests that test all the examples.
- Coverage tested for each PR via Codecov & cdash.
- **Upcoming feature (FairMQ 1.4):** Run tests via DDS to increase robustness against used ports.

Tracking issue <https://github.com/FairRootGroup/FairMQ/issues/75>

codecov bot commented 23 hours ago

## Codecov Report

Merging #125 into dev will increase coverage by 3.14%.  
The diff coverage is 100%.

	Coverage	Diff	
Files	66	66	
Lines	4014	3914	-100
+ Hits	2889	2940	+51
+ Misses	1125	974	-151

https://cdash.gsi.de/index.php?project=FairMQ

Thursday, November 29 2018 13:43:40

Dashboard Calendar Previous Current Project

No file changed as of Thursday, November 29 2018 - 01:00 UTC

20 hours ago: 1 test failed on codecov-gcc8.1.0-FairMQ\_PR-125  
 22 hours ago: 1 warning introduced on codecov-gcc8.1.0-FairMQ\_PR-121  
 22 hours ago: 4 tests not run on codecov-gcc8.1.0-FairMQ\_dev  
 22 hours ago: 21 tests failed on codecov-gcc8.1.0-FairMQ\_dev  
 22 hours ago: 1 warning introduced on codecov-gcc8.1.0-FairMQ\_dev

See full feed

### Nightly

1 build

Site	Build Name	Configure		Build		Test			Start Time	Labels
		Error	Warn	Error	Warn	Not Run	Fail	Pass		
lxbk0199.gsi.de	Debian8-x86_64-gcc8.1.0-FairMQ_dev	0	0	0	0	0	0	41 <sup>+1</sup> <sub>-1</sub>	11 hours ago	(none)

### Experimental

3 builds

Site	Build Name	Configure		Build		Test			Start Time	Labels
		Error	Warn	Error	Warn	Not Run	Fail	Pass		
lxbk0199.gsi.de	codecov-gcc8.1.0-FairMQ_master	0	0	0	0	0	0	41	1 hour ago	(none)
lxbk0199.gsi.de	alfa_ci-gcc8.1.0-FairMQ_master	0	0	0	0	0	0	41	1 hour ago	(none)
lxbk0199.gsi.de	Debian8-x86_64-gcc8.1.0-FairMQ_dev	0	0	0	0	0	0	41 <sup>+1</sup> <sub>-1</sub>	11 hours ago	(none)

### Coverage

Site	Build Name	Percentage	LOC Tested	LOC Untested	Date	Labels
lxbk0199.gsi.de	Debian8-x86_64-gcc8.1.0-FairMQ_dev	81.43% <sup>+1.62%</sup>	3485 <sup>+50</sup>	795 <sup>-74</sup>	11 hours ago	(none)
lxbk0199.gsi.de	codecov-gcc8.1.0-FairMQ_master	81.4%	3484	796	1 hour ago	(none)

Kitware CDash 2.5.0 © Kitware | Report problems | Data privacy protection | Impressum | 0.34s

# Plans (FairMQ v1.4.x)

Rework polling interface/implementation.

- Current polling interface (FairMQPoller) is transport-specific and thus makes it difficult to combine different transports into the same poll set.
  - For example, OnData with different transports will use an extra thread for a poller of each transport.
  - Additionally the poll API of ZeroMQ/nanomsg is  $O(n)$ , which can lead to decreased performance with large numbers of sockets.
- A common polling interface based on **boost::asio** is feasible, that takes a file descriptor from each poll entry and initiates async reads on the file descriptors that can be then handled asynchronously, without iterating over all sockets on each trigger ( $O(1)$ ).

Tracking issue

<https://github.com/FairRootGroup/FairMQ/issues/7>

Further plans for v1.4 release

<https://github.com/FairRootGroup/FairMQ/milestone/3>

# Issue reports & PRs

Included in v1.3.7

<b>PR #93</b>	<b>Adopt FairMQMessage backed memory resource collection from AliceO2</b>	<b>Mikolaj (WP1)</b>
<b>Issue #82</b>	<b>Provide a sleep method that is interruptible (e.g. by a state change)</b>	<b>Giulio (WP8)</b>
<b>Issue #109</b>	<b>Bad configuration transitions to RUNNING before jumping back to READY</b>	<b>Teo (WP8)</b>
<b>PR #114</b>	<b>Use a safer random generator from boost</b>	<b>Teo (WP8)</b>
<b>Issue #83</b>	<b>Deadlock at device shutdown</b>	<b>Sandro (WP12)</b>
<b>Issue #95</b>	<b>Race conditions when throwing an exception</b>	<b>Giulio (WP4)</b>

**Thanks everyone for your contributions and reports!**

# Thank You for Your Attention

**FairMQ**

<https://github.com/FairRootGroup/FairMQ/>

**Latest release notes (v1.3.7)**

<https://github.com/FairRootGroup/FairMQ/releases/tag/v1.3.7>

**Milestones for v1.4 release**

<https://github.com/FairRootGroup/FairMQ/milestone/3>

**Docs**

<https://github.com/FairRootGroup/FairMQ/blob/master/README.md#documentation>

**API docs**

<https://fairrootgroup.github.io/FairMQ/latest>

**CDash**

<https://cdash.gsi.de/index.php?project=FairMQ>