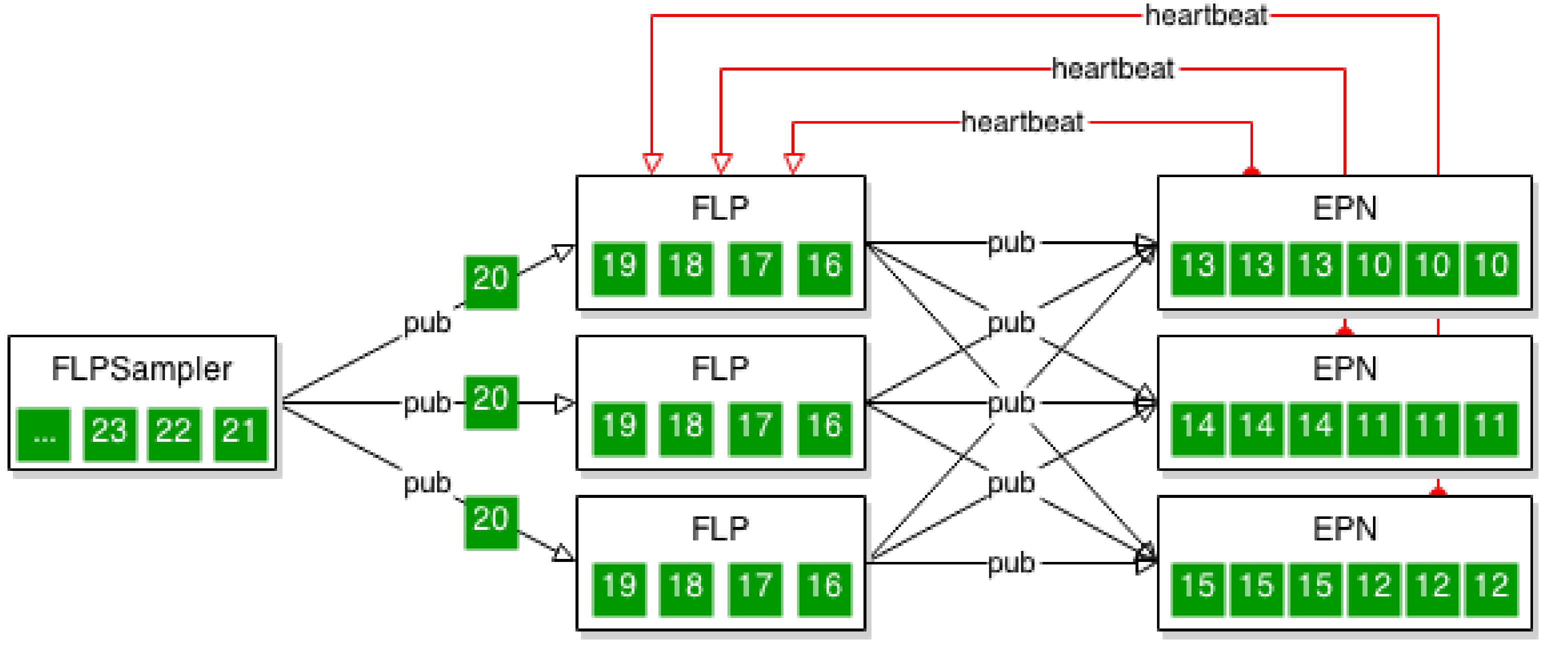
Implementation of the staggered data transfer

Alexey Rybalchenko

Data Distribution

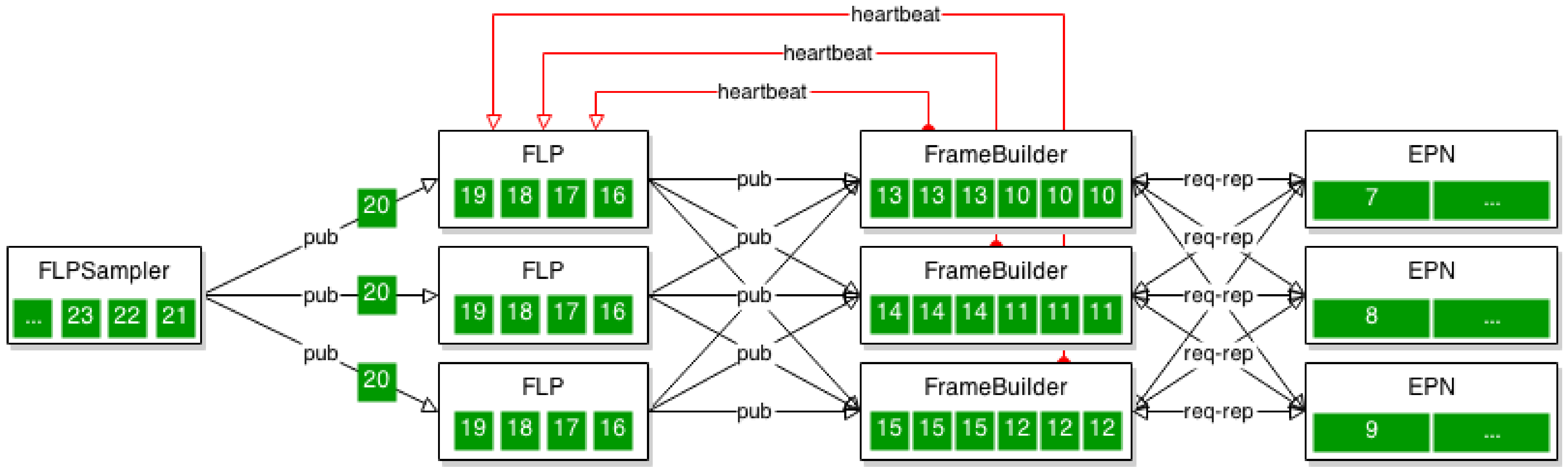


Payload for tests: ZeroMQ multipart message:

`Event ID`	`Data of configurable size`

FLP decides where to send the payload from the event ID: `eventID % numOfOutputs`

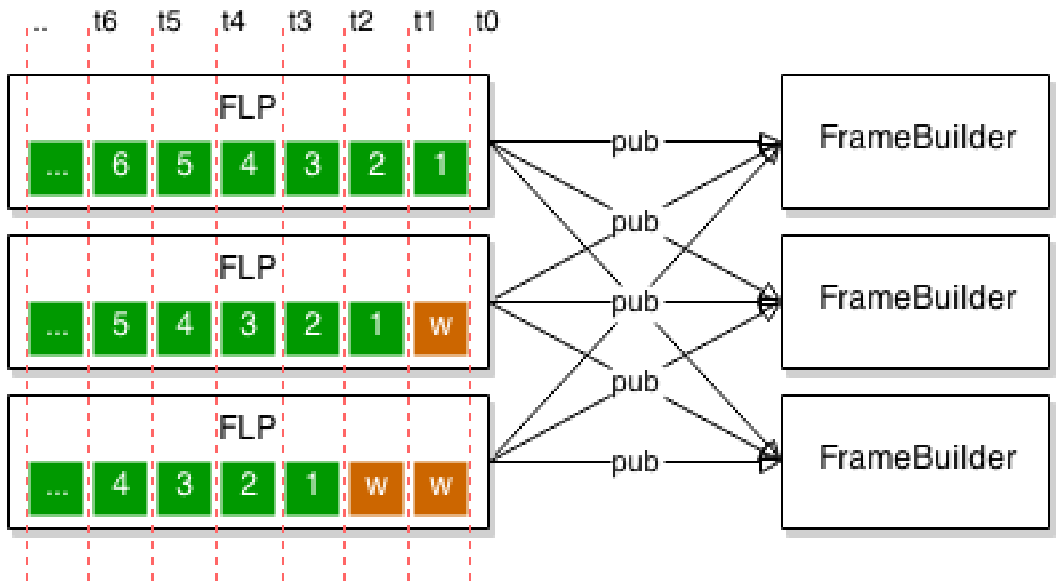
Frame Builders



FrameBuilder: separate device to collect event parts together.

Available EPNs request work from FrameBuilders with request-reply

Staggered Transfer

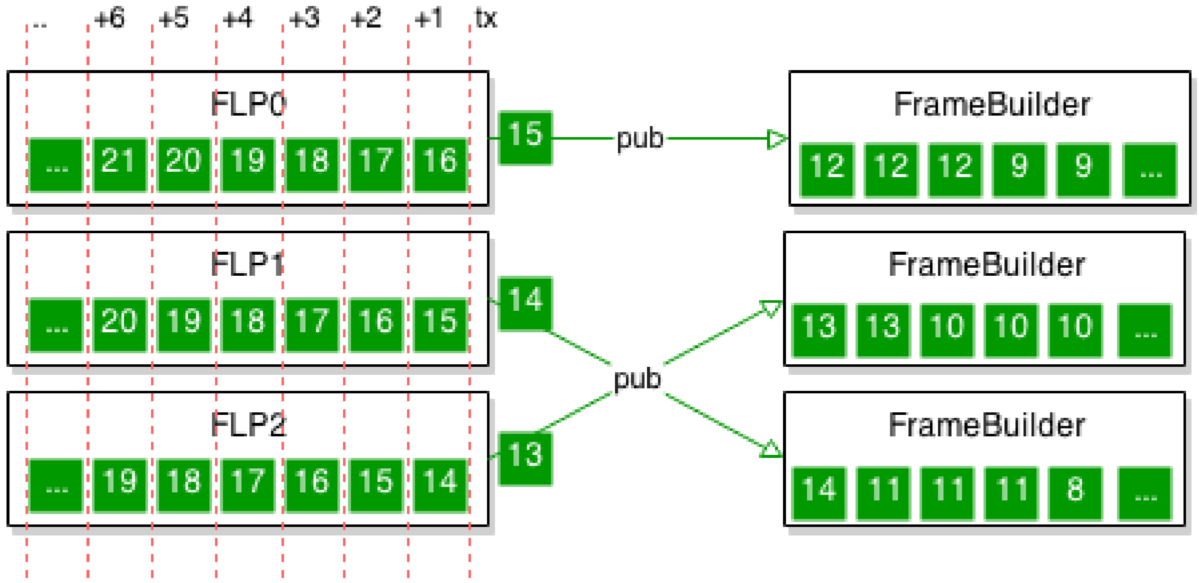


All FLPs sending at the same time to one EPN/FrameBuilder would overload the network path.

First step: Delay first sending of some FLPs by an offset, store pending messages in a buffer.

Buffer size: <# of FLPs> FairMQMessages.

Staggered Transfer



When the buffers are full all FLPs can send at the same time, payloads go to different EPNs/Frame Builders.

Buffer on FLPs is a „zero-copy“ buffer – no additional overhead from copying.
(keeps message until its sending turn,
deallocation done automatically by ZeroMQ after the queued message is sent out).

Synchronization depends on the simultaneous receiving of the payload by FLP.

Current progress available on GitHub:
https://github.com/rbx/AliceO2/tree/FLP2EPN-distributed

FairMQ Updates

boost::program_options for more flexible command line parameters handling.

→Allows easy use of required parameters, default values, repeated parameters (inputs/outputs), automatic --help generation, reading parameters both from command line and/or config file.

Example in: https://github.com/rbx/AliceO2/tree/FLP2EPN-distributed

Command socket: used to change FairMQDevice states (init, run, pause, etc.)

Further uses? Query of the current state, command propagation (e.g.: FLP stop -> EPN stop)