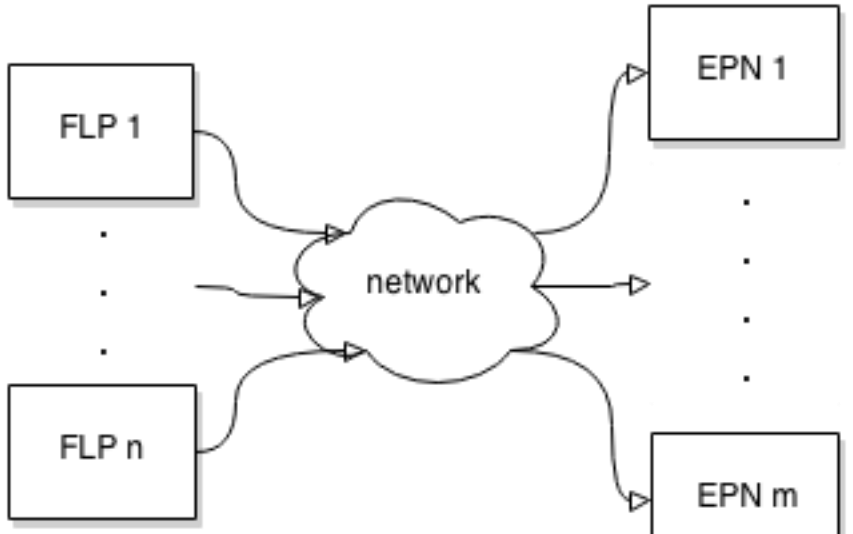
„staggered“ data transfer shaping
development and first results

Alexey Rybalchenko

ALICE Offline week : 2014-11-20

FLP to EPN Scenario

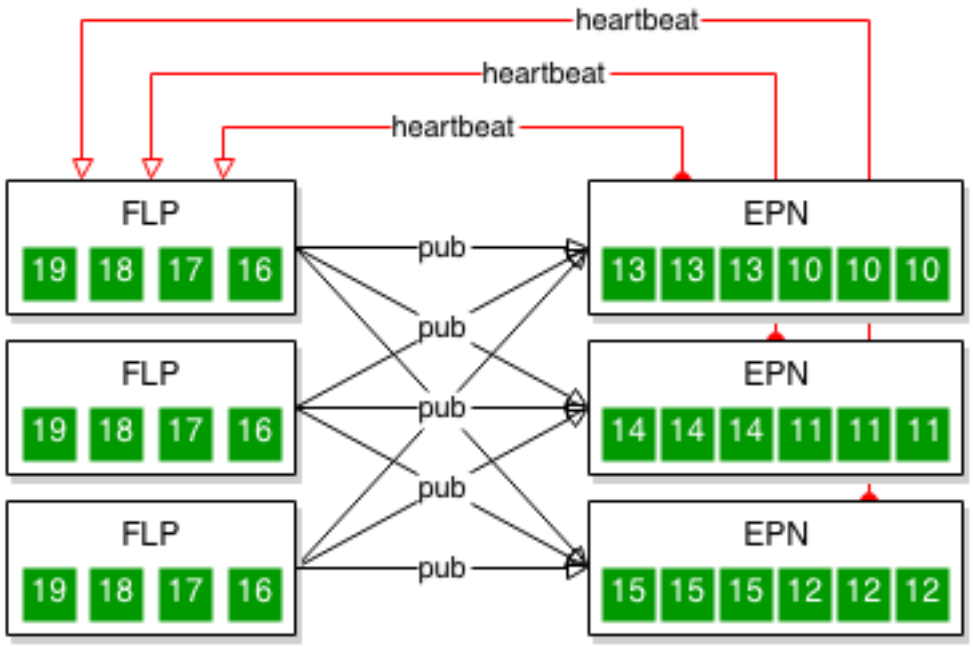


For the Run 3, the ALICE detector will produce over 1 TB/s output.

This data have to be distributed from about 250 Front Level Processors to O(1000) Event Processing Nodes.

FairMQ is proposed as a transport solution between FLPs and EPNs.
FairMQ enables asynchronous communication via messages between processes/nodes.
The underlying transport of FairMQ is implemented via ZeroMQ or nanomsg libraries,
with the transport interface allowing implementation with other techniques.

Data Distribution



FLP decides where to send the payload from the timeframe ID: `TimeframeID % numOfOutputs` (EPN availability is tracked with the heartbeat received within timeout window).

Current status: prototype of distribution and collecting of timeframes on the EPNs is working!

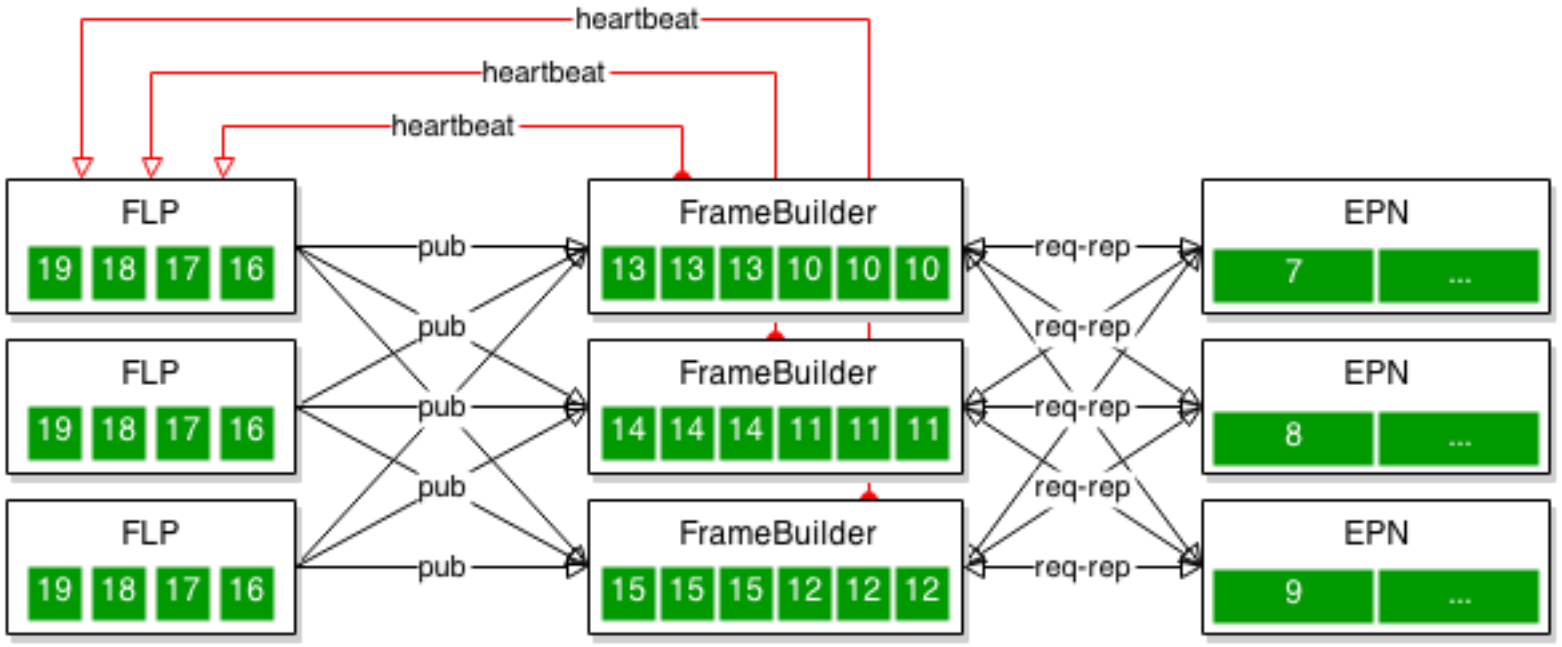
This can be replaced with EPN scheduling (Sylvain Chapeland), where the scheduling service will provide an EPN ID to which to send a given Timeframe ID.

Current step: integrate the service with the FLP2EPN prototype. Test/compare performance.

Payload for tests: FairMQ multipart message:

Timeframe ID	Data of configurable size

Alternative: Frame Builders

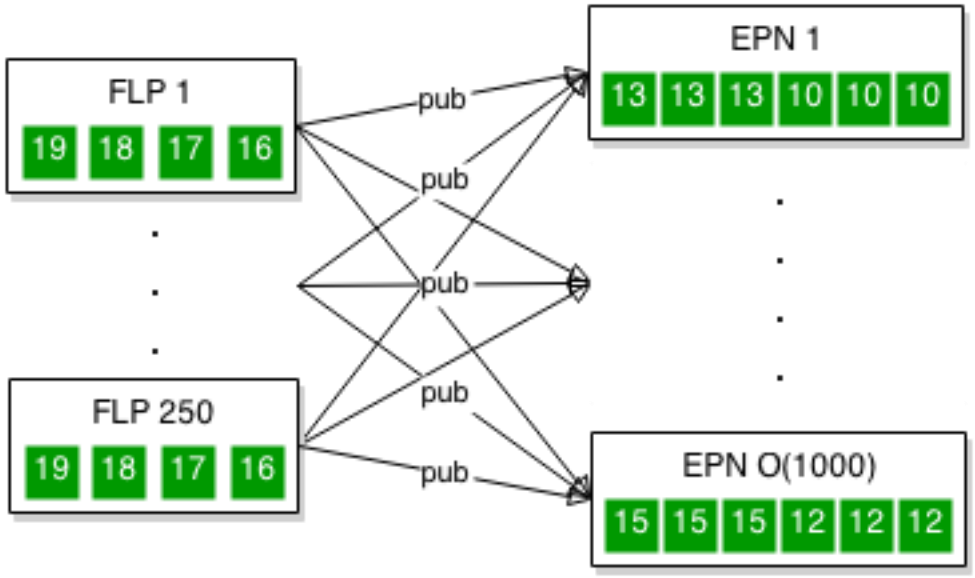


FrameBuilder: separate device to collect timeframes together.

Available EPNs request work from FrameBuilders via request-reply pattern.

Current status: in development.

FLP2EPN Network Load

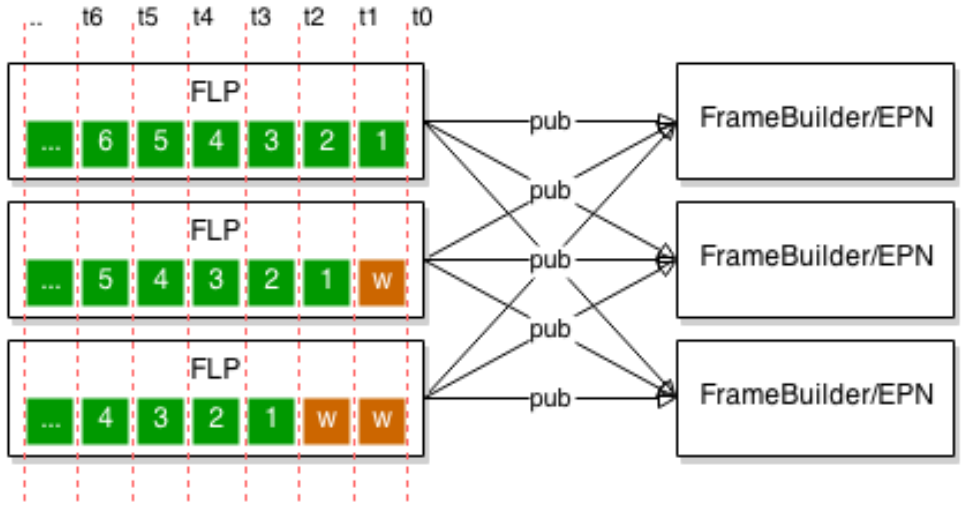


All FLPs sending at the same time to a single EPN could overload the network path.

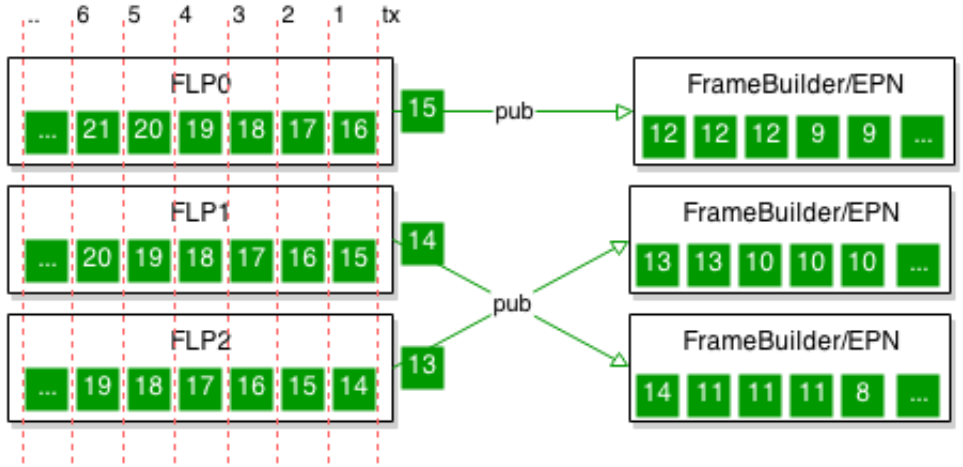
FLPs should operate at maximum performance, so synchronization between them should be avoided/minimized.

First version of a data transfer shaping algorithm is implemented to minimize the contention.

Staggered Transfer



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



Approach: Delay first sending of some FLPs by an offset, storing pending messages in a buffer.

Buffer: std::queue (FIFO) of FairMQMessage pointers.
(no copying/moving of the message content,
automatic deallocation after FairMQ has sent out the data)

Maximum buffer size: <# of FLPs> FairMQMessages.

Offset/Buffer: configurable!
The optimal value for the offset depends on the use case.

Status: configurable offset/buffer is working,
(current progress available on GitHub:
https://github.com/rbx/AliceO2/tree/FLP2EPN-distributed)
now testing & measuring performance.
HLT dev cluster: 10 nodes connected via 40Gbit Infiniband.
Current test setups: 16 FLPs -> 16 EPNs, 64 FLPs -> 64 EPNs.

Very early measurements

Payload for tests: FairMQ multipart message:

16 FLPs (4 nodes) -> 16 EPNs (4 nodes)

```
[19:23:57][DEBUG] #EPN09.sub.0: 132 msg/s, 629.426 MB/s
[19:23:58][DEBUG] #EPN09.sub.0: 122 msg/s, 581.742 MB/s
[19:23:59][DEBUG] #EPN09.sub.0: 122 msg/s, 581.742 MB/s
[19:24:00][DEBUG] #EPN09.sub.0: 132 msg/s, 629.426 MB/s
[19:24:01][DEBUG] #EPN09.sub.0: 110 msg/s, 524.521 MB/s
[19:24:02][DEBUG] #EPN09.sub.0: 114 msg/s, 543.595 MB/s
[19:24:03][DEBUG] #EPN09.sub.0: 120 msg/s, 572.205 MB/s
[19:24:04][DEBUG] #EPN09.sub.0: 116 msg/s, 553.132 MB/s
[19:24:05][DEBUG] #EPN09.sub.0: 118 msg/s, 562.668 MB/s
[19:24:06][DEBUG] #EPN09.sub.0: 122 msg/s, 581.742 MB/s
[19:24:07][DEBUG] #EPN09.sub.0: 112 msg/s, 534.058 MB/s
[19:24:08][DEBUG] #EPN09.sub.0: 112 msg/s, 534.058 MB/s
[19:24:09][DEBUG] #EPN09.sub.0: 122 msg/s, 581.742 MB/s
[19:24:10][DEBUG] #EPN09.sub.0: 114 msg/s, 543.595 MB/s
[19:24:11][DEBUG] #EPN09.sub.0: 120 msg/s, 572.205 MB/s
```

~567 MB/s x 4 processes per node ≈
~2271 MB/s throughput / (on 1 EPN)

64 FLPs (4 nodes) -> 64 EPNs (4 nodes)

```
[21:14:40][DEBUG] #EPN64.sub.0: 24 msg/s, 114.441 MB/s
[21:14:41][DEBUG] #EPN64.sub.0: 18 msg/s, 85.8308 MB/s
[21:14:42][DEBUG] #EPN64.sub.0: 34 msg/s, 162.125 MB/s
[21:14:43][DEBUG] #EPN64.sub.0: 46 msg/s, 219.345 MB/s
[21:14:44][DEBUG] #EPN64.sub.0: 24 msg/s, 114.441 MB/s
[21:14:45][DEBUG] #EPN64.sub.0: 24 msg/s, 114.441 MB/s
[21:14:46][DEBUG] #EPN64.sub.0: 28 msg/s, 133.515 MB/s
[21:14:47][DEBUG] #EPN64.sub.0: 50 msg/s, 238.419 MB/s
[21:14:48][DEBUG] #EPN64.sub.0: 36 msg/s, 171.662 MB/s
[21:14:49][DEBUG] #EPN64.sub.0: 32 msg/s, 152.588 MB/s
[21:14:50][DEBUG] #EPN64.sub.0: 24 msg/s, 114.441 MB/s
[21:14:51][DEBUG] #EPN64.sub.0: 42 msg/s, 200.272 MB/s
[21:14:52][DEBUG] #EPN64.sub.0: 34 msg/s, 162.125 MB/s
[21:14:53][DEBUG] #EPN64.sub.0: 32 msg/s, 152.588 MB/s
[21:14:54][DEBUG] #EPN64.sub.0: 28 msg/s, 133.515 MB/s
```

~150 MB/s x 16 processes per node ≈
~2413 MB/s throughput (on 1 EPN)

msg/s reports double numbers of msgs because it is currently counting msg parts (multipart)