

MQ

Day 1

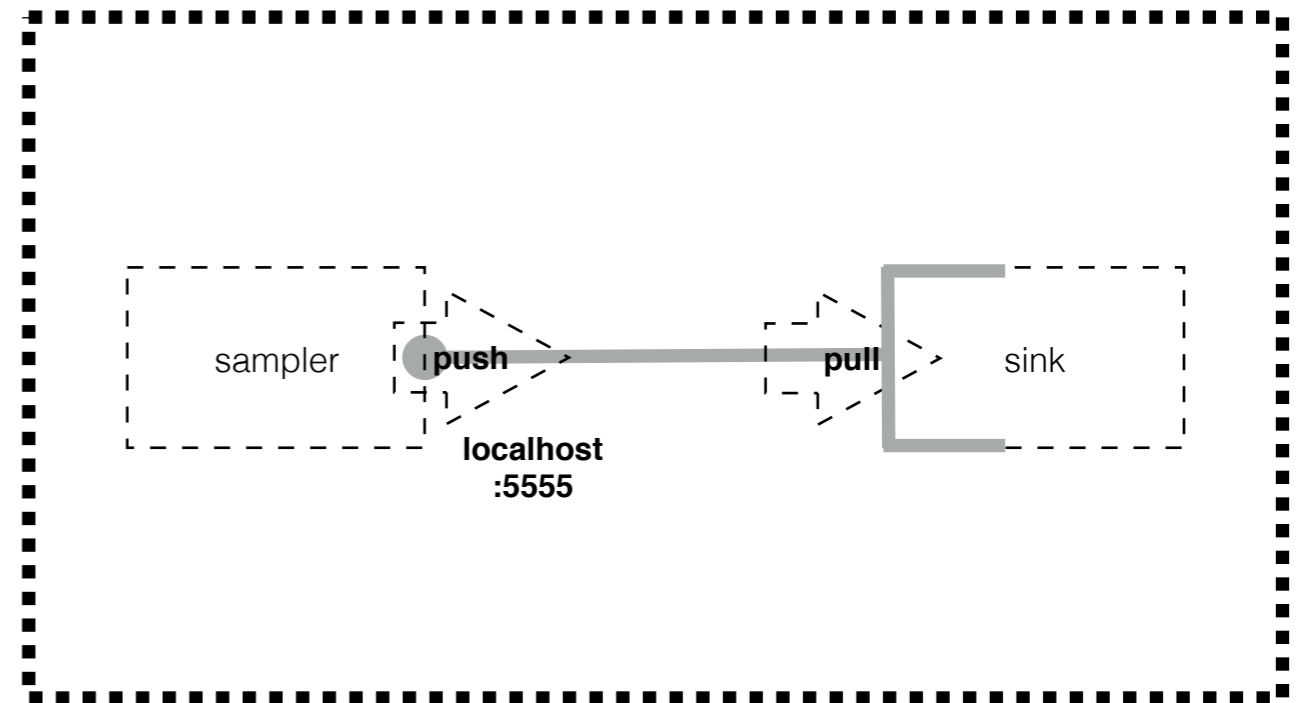
- implement simple devices and topologies
- learn what 'send' is
- learn about 'OnData'
- difference between bind and connect

- shell script to start multiple devices?
- introduce proxy?

- /// kind of extended MQ/example1

Day 1. Exercise 1

- create simple sampler and sink devices (send and receive string), together with executables
- create CMakeLists.txt (library, executables)
- create topology consisting of two devices:
 - sampler1 (binding transport channel to push on this channel)
 - sink1 (connecting to said channel to pull from this channel)



Download:

```
panda@panda-workshop:~/workshop/PandaRoot-trunk$ wget http://web-docs.gsi.de/~karabowi/thailand/MQ\_basics.tgz
```

```
panda@panda-workshop:~/workshop/PandaRoot-trunk$ ls -l MQ_basics.tgz  
-rw-r--r--  1 karabowi  staff  1736 Jun 27 08:54 MQ_basics.tgz
```

```
panda@panda-workshop:~/workshop/PandaRoot-trunk$ tar xvzf MQ_basics.tgz  
x MQ_basics/  
x MQ_basics/CMakeLists.txt  
x MQ_basics/devices/  
x MQ_basics/devices/PndMQ1Sampler.cxx  
x MQ_basics/devices/PndMQ1Sink.h  
x MQ_basics/devices/PndMQ1Sink.cxx  
x MQ_basics/devices/PndMQ1Sampler.h  
x MQ_basics/scripts/  
x MQ_basics/run/  
x MQ_basics/run/runPndMQ1Sampler.cxx  
x MQ_basics/run/runPndMQ1Sink.cxx  
x MQ_basics/options/  
x MQ_basics/options/MQ1_sampler_sink.json
```

```
panda@panda-workshop:~/workshop/PandaRoot-trunk$ mv MQ_basics MQ_samp_sink
```

Implement

- Edit sampler device code (MQ_samp_sink/devices/PndMQ1Sampler) and make it send a string:

```
bool PndMQ1Sampler::ConditionalRun()
{
    std::this_thread::sleep_for(std::chrono::milliseconds(1000));
    string* text = new string("Hello");
    FairMQMessagePtr msg(NewMessage(const_cast<char*>(text->c_str()),
    LOG(INFO) << "Sending \"" << text->c_str() << "\"");
    if ( send(msg, "data") < 0 )
    {
        return false;
    }
    return true;
}
```

wait 1s

create string

create message

send message

stop running if sending failed

continue running device

- **ConditionalRun()** is a virtual function of **FairMQDevice**. It is used in the devices that do not have input channel.
- **ConditionalRun()** is executed in a loop as long as it returns **true**.

Implement

- Edit sink device code (MQ_samp_sink/devices/PndMQ1Sink) and print a received message:

```
PndMQ1Sink::PndMQ1Sink()
{
    OnData("data", &PndMQ1Sink::HandleData);
}

bool PndMQ1Sink::HandleData(FairMQMessagePtr& msg, int /*index*/)
{
    LOG(INFO) << "Received: \"" << string(static_cast<char*>(msg->GetData()), msg->GetSize()) << "\"";
    return true;
}
```

connect a HandleData function to the "data" channel

function implementation

print the data from the received message

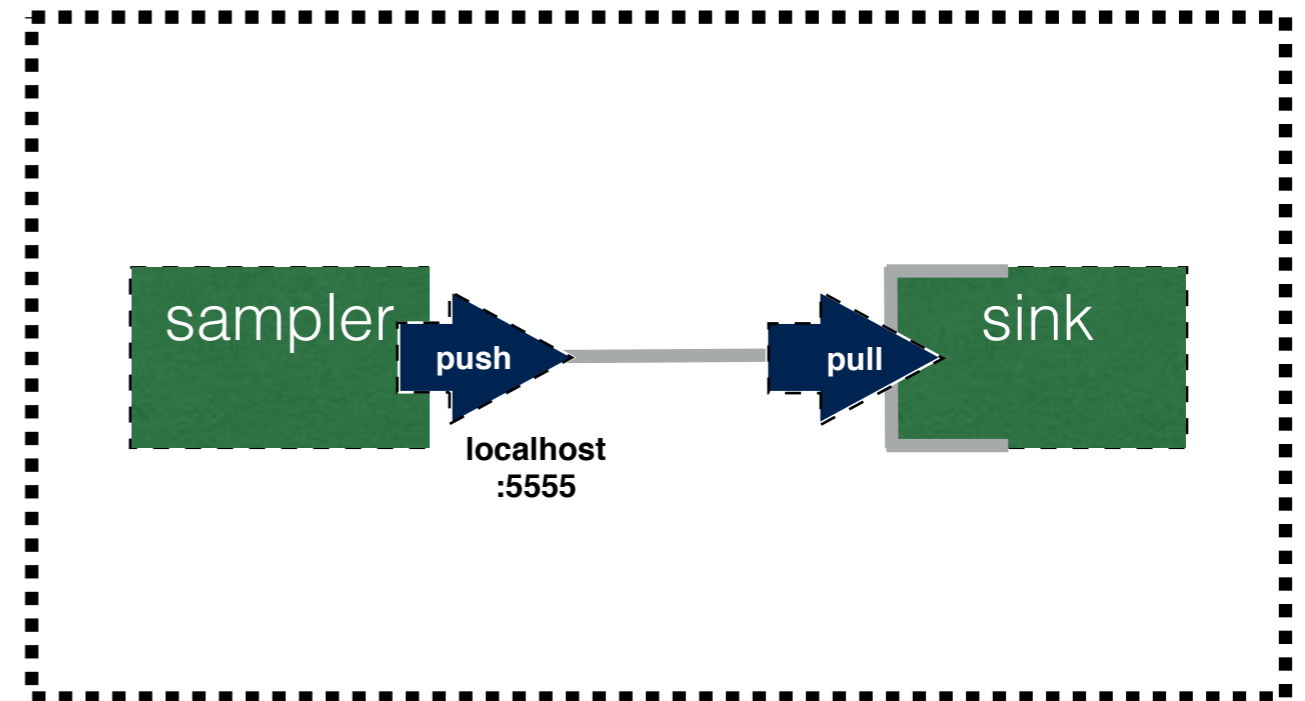
- Connect receiving channel to the appropriate function in the constructor of the device, using OnData("channelName",&functionName).
- Implement function to process the received message.

Day 1.Excercise1

- add_subdirectory(MQ_samp_sink) to the main CMakeLists.txt;
- compile;
- run:

```
build$ ./bin/mq1-sink --id sink1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_sink/options/MQ1.json
```

```
build$ ./bin/mq1-sampler --id sampler1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_sink/options/MQ1.json
```



Improve

- Edit executable code (MQ_samp_sink/run/runPndMQ1Sampler.cxx) to add two option-variables:

```
#include "runFairMQDevice.h"
#include "PndMQ1Sampler.h"

namespace bpo = boost::program_options;

void addCustomOptions(bpo::options_description& options)
{
    options.add_options()
        ("text", bpo::value<std::string>()->default_value("Hello"), "Text to sendout")
        ("delay", bpo::value<int>()->default_value(1000), "Delay in milliseconds");
}

FairMQDevicePtr getDevice(const FairMQProgOptions& /*config*/)
{
    return new PndMQ1Sampler();
}
```

- The device has to run as a separate process.
- To ease creation of the executable, a header runFairMQDevice.h containing generic main() function has been provided.
- In the user code, only extra options and instructions to run a specific device have to be added.

Improve

- Edit sampler device code to read the value of the global variables from the FairMQProgOptions:

```
void PndMQ1Sampler::InitTask()  
{  
    fText = fConfig->GetValue<string>("text");  
    fDelay = fConfig->GetValue<int>("delay");  
}
```

- use the fText and fDelay in the ConditionalRun()
- compile
- run and use the options from the command line

Comment

- Edit the topology description file

```
{
  "fairMQOptions": {
    "devices": [
      {
        "id": "sampler1",
        "channels": [
          {
            "name": "data",
            "sockets": [
              {
                "type": "push",
                "method": "bind",
                "address": "tcp://*:5555",
                "sndBufSize": 1000,
                "rcvBufSize": 1000,
                "rateLogging": 0
              }
            ]
          }
        ]
      },
      {
        "id": "sink1",
        "channels": [
          {
            "name": "data",
            "sockets": [
              {
                "type": "pull",
                "method": "connect",
                "address": "tcp://localhost:5555",
                "sndBufSize": 1000,
                "rcvBufSize": 1000,
                "rateLogging": 0
              }
            ]
          }
        ]
      }
    ]
  }
}
```

defines device with id sampler1
with one channel named "data"

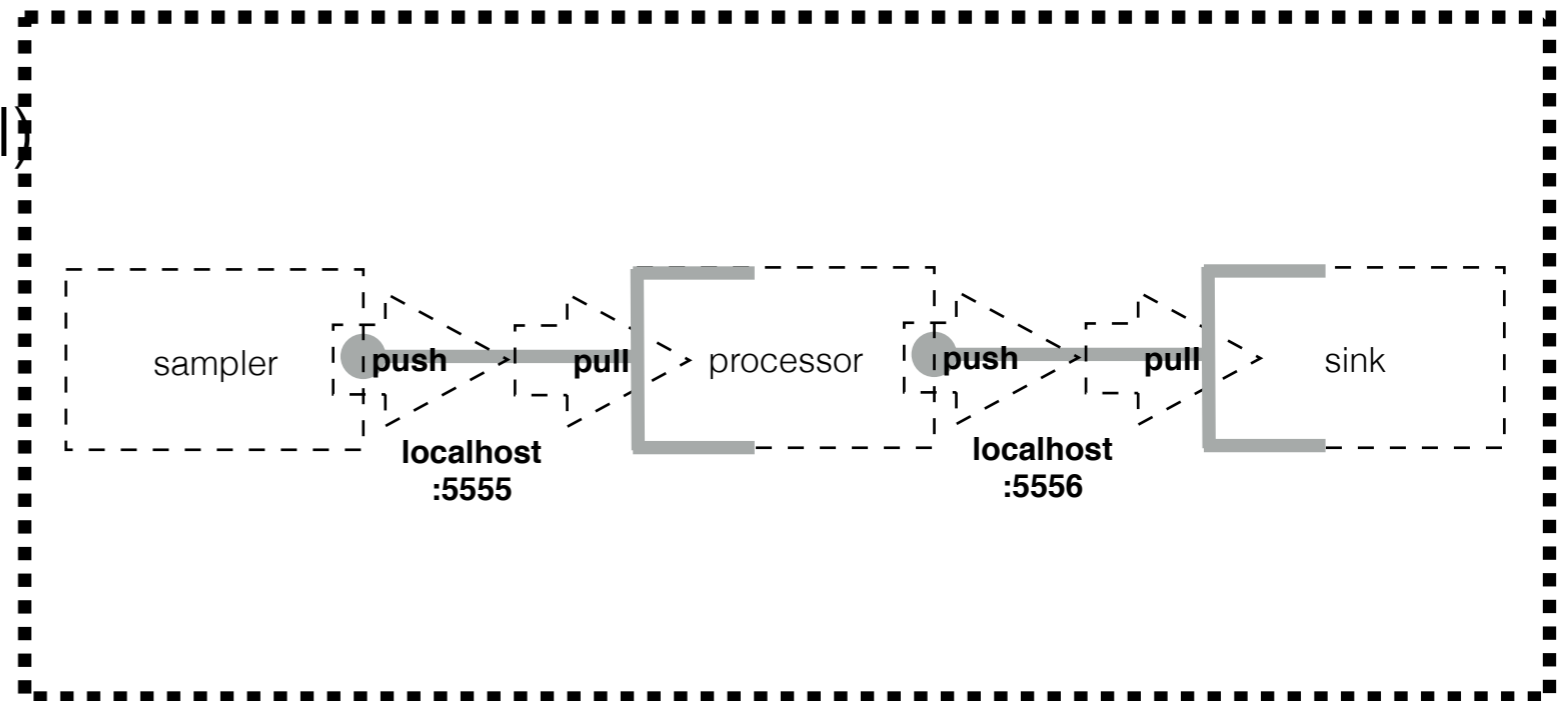
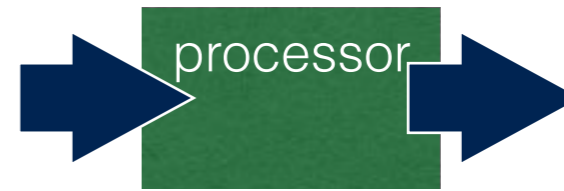
with one socket
the socket will bind on port 5555, and will push data
the buffer size is 1000 messages
the logging rate is 0 (no logging)

defines device with id sink1
with one channel named "data"

with one socket
the socket will connect to localhost port 5555, and will pull data
the buffer size is 1000 messages
the logging rate is 0 (no logging)

Day 1. Exercise 2

- create simple processor to process string, together with executable
- create CMakeLists.txt (library, executables)
- create topology consisting of three devices:
 - sampler1 (binding transport channel to push on this channel)
 - proc1 (connect to sampler channel, bind new channel)
 - sink1 (connecting to processor channel to pull from this channel)



Implement

- The processor creates output message based on input message, fe:

```
bool PndMQ2Proc::HandleData(FairMQMessagePtr& msg, int /*index*/)
{
    fText = string(static_cast<char*>(msg->GetData()), msg->GetSize());
    fText.insert(0, fId);
    fText.insert(0, "processed_");
}
```

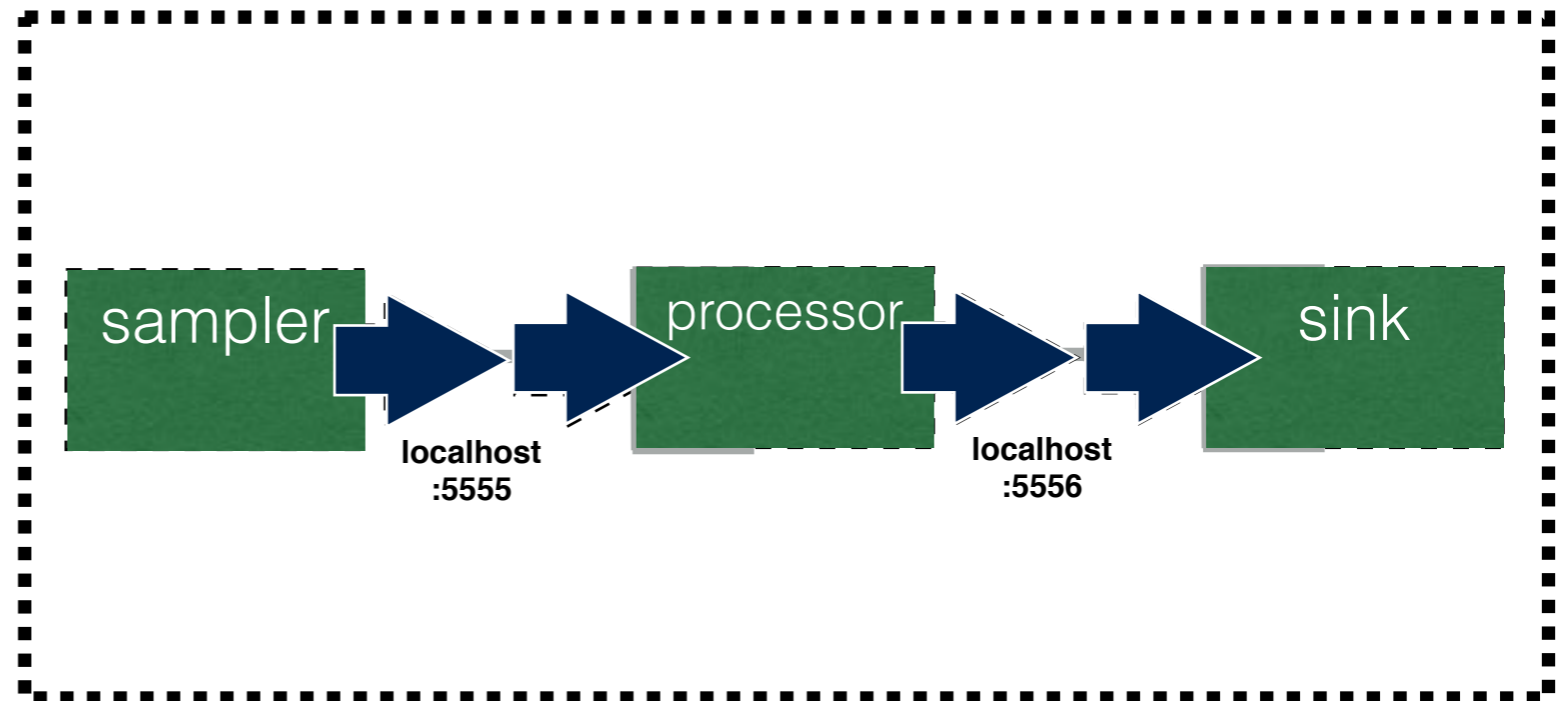
Day 1. Exercise 2

- run:

```
build$ ./bin/mq1-sink --id sink1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```

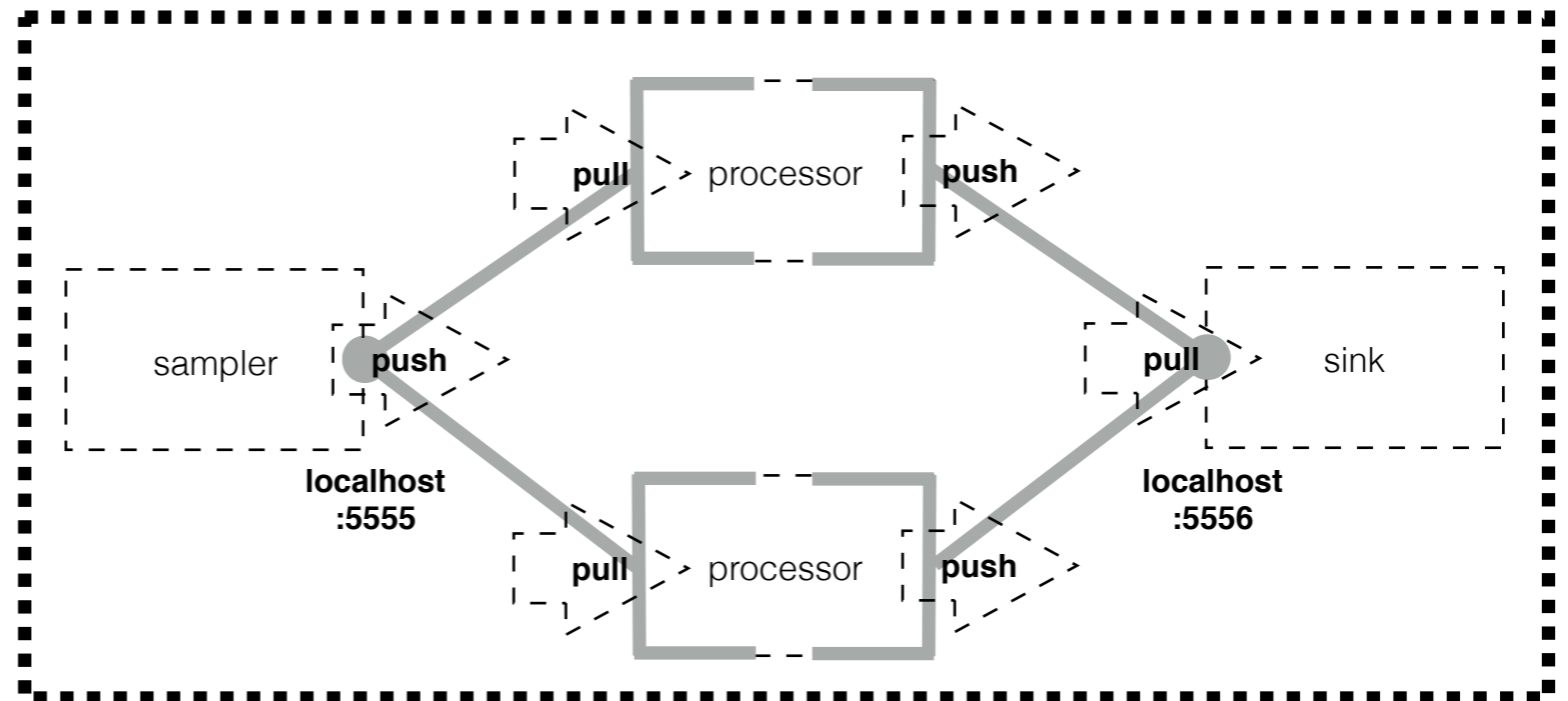
```
build$ ./bin/mq2-proc --id proc1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```

```
build$ ./bin/mq1-sampler --id sampler1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```



Day 1. Exercise 3

- modify topology to add one more processor:
- sampler1 (binding transport channel to push on this channel)
- proc1 (connect to sampler and sink channels)
- proc1 (connect to sampler and sink channels)
- sink1 (bind sink channel to pull from this channel)



Day 1. Exercise 3

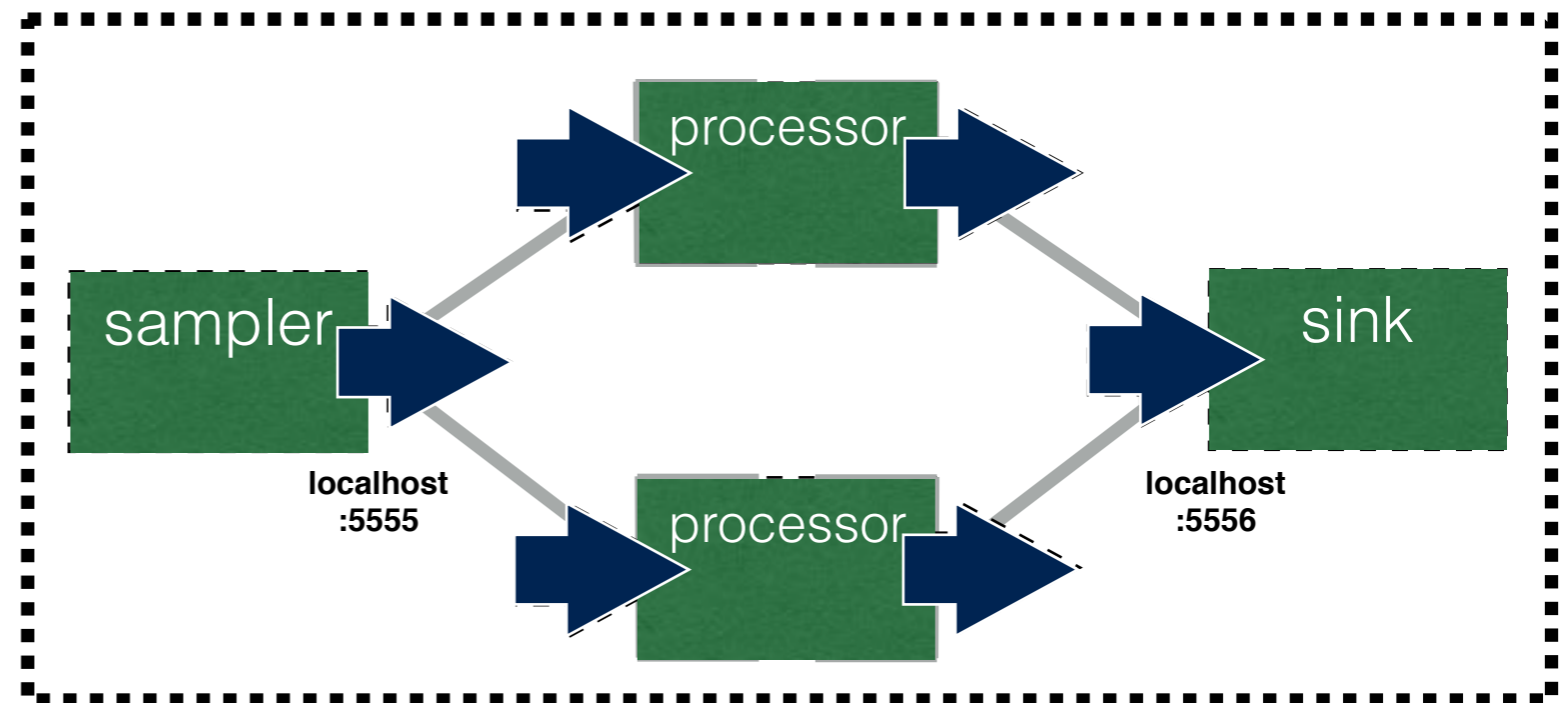
- run & play:

```
build$ ./bin/mq1-sink --id sink1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```

```
build$ ./bin/mq2-proc --id proc1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```

```
build$ ./bin/mq2-proc --id proc2 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```

```
build$ ./bin/mq1-sampler --id sampler1 --mq-config ~/workshop/PandaRoot_trunk/MQ_samp_proc_sink/options/MQ2.json
```



Day 1 Script

```
X) mq1-sampler
12:15:22 [DEBUG] Transport: Using ZeroMQ library, version: 4.1.3
12:15:22 [DEBUG] Adding 'zeromq' transport to the device.
12:15:22 [STATE] Entering INITIALIZING DEVICE state
12:15:22 [DEBUG] Validating channel 'data[0]'... VALID
12:15:22 [DEBUG] data[0]: using default transport
12:15:22 [DEBUG] Attached channel data[0] to tcp://*:5555 (bind)
12:15:22 [STATE] Entering DEVICE READY state
12:15:22 [DEBUG] Init time (CPU) : 0.70 ms
12:15:22 [DEBUG] Init time (wall): 0.64 ms
12:15:22 [STATE] Entering INITIALIZING TASK state
12:15:22 [STATE] Entering READY state
12:15:22 [STATE] Entering RUNNING state
12:15:22 [INFO] DEVICE: Running...
12:15:22 [INFO] Use keys to control the state machine:
12:15:22 [INFO] [h] help, [p] pause, [r] run, [s] stop, [t] reset task, [d] reset device, [q] end, [i] init task, [I] init device
12:15:23 [INFO] Sending "Hello"
12:15:24 [INFO] Sending "Hello"
12:15:25 [INFO] Sending "Hello"
12:15:26 [INFO] Sending "Hello"
12:15:27 [INFO] Sending "Hello"
12:15:28 [INFO] Sending "Hello"

X) mq1-sink
12:15:22 [DEBUG] Transport: Using ZeroMQ library, version: 4.1.3
12:15:22 [DEBUG] Adding 'zeromq' transport to the device.
12:15:22 [STATE] Entering INITIALIZING DEVICE state
12:15:22 [DEBUG] Validating channel 'data[0]'... VALID
12:15:22 [DEBUG] data[0]: using default transport
12:15:22 [DEBUG] Attached channel data[0] to tcp://*:5555 (bind)
12:15:22 [STATE] Entering DEVICE READY state
12:15:22 [DEBUG] Init time (CPU) : 0.71 ms
12:15:22 [DEBUG] Init time (wall): 1.25 ms
12:15:22 [STATE] Entering INITIALIZING TASK state
12:15:22 [STATE] Entering READY state
12:15:22 [STATE] Entering RUNNING state
12:15:22 [INFO] DEVICE: Running...
12:15:22 [INFO] Use keys to control the state machine:
12:15:22 [INFO] [h] help, [p] pause, [r] run, [s] stop, [t] reset task, [d] reset device, [q] end, [i] init task, [I] init device
12:15:23 [INFO] Received: "processed_processor2Hello"
12:15:24 [INFO] Received: "processed_processor1Hello"
12:15:25 [INFO] Received: "processed_processor2Hello"
12:15:26 [INFO] Received: "processed_processor1Hello"
12:15:27 [INFO] Received: "processed_processor2Hello"
12:15:28 [INFO] Received: "processed_processor1Hello"

X) mq2-sampler
12:15:22 [DEBUG] Transport: Using ZeroMQ library, version: 4.1.3
12:15:22 [DEBUG] Adding 'zeromq' transport to the device.
12:15:22 [STATE] Entering INITIALIZING DEVICE state
12:15:22 [DEBUG] Validating channel 'dataOut[0]'... VALID
12:15:22 [DEBUG] dataOut[0]: using default transport
12:15:22 [DEBUG] Attached channel dataOut[0] to tcp://localhost:5556 (connect)
12:15:22 [DEBUG] Validating channel 'dataIn[0]'... VALID
12:15:22 [DEBUG] dataIn[0]: using default transport
12:15:22 [DEBUG] Attached channel dataIn[0] to tcp://localhost:5555 (connect)
12:15:22 [STATE] Entering DEVICE READY state
12:15:22 [DEBUG] Init time (CPU) : 1.60 ms
12:15:22 [DEBUG] Init time (wall): 55.88 ms
12:15:22 [STATE] Entering INITIALIZING TASK state
12:15:22 [STATE] Entering READY state
12:15:22 [STATE] Entering RUNNING state
12:15:22 [INFO] DEVICE: Running...
12:15:22 [INFO] Use keys to control the state machine:
12:15:22 [INFO] [h] help, [p] pause, [r] run, [s] stop, [t] reset task, [d] reset device, [q] end, [i] init task, [I] init device
12:15:24 [INFO] Sending "processed_processor1Hello"
12:15:25 [INFO] Sending "processed_processor2Hello"
12:15:26 [INFO] Sending "processed_processor1Hello"

X) mq2-sink
12:15:22 [DEBUG] Transport: Using ZeroMQ library, version: 4.1.3
12:15:22 [DEBUG] Adding 'zeromq' transport to the device.
12:15:22 [STATE] Entering INITIALIZING DEVICE state
12:15:22 [DEBUG] Validating channel 'dataOut[0]'... VALID
12:15:22 [DEBUG] dataOut[0]: using default transport
12:15:22 [DEBUG] Attached channel dataOut[0] to tcp://localhost:5556 (connect)
12:15:22 [DEBUG] Validating channel 'dataIn[0]'... VALID
12:15:22 [DEBUG] dataIn[0]: using default transport
12:15:22 [DEBUG] Attached channel dataIn[0] to tcp://localhost:5555 (connect)
12:15:22 [STATE] Entering DEVICE READY state
12:15:22 [DEBUG] Init time (CPU) : 1.15 ms
12:15:22 [DEBUG] Init time (wall): 52.21 ms
12:15:22 [STATE] Entering INITIALIZING TASK state
12:15:22 [STATE] Entering READY state
12:15:22 [STATE] Entering RUNNING state
12:15:22 [INFO] DEVICE: Running...
12:15:22 [INFO] Use keys to control the state machine:
12:15:22 [INFO] [h] help, [p] pause, [r] run, [s] stop, [t] reset task, [d] reset device, [q] end, [i] init task, [I] init device
12:15:23 [INFO] Received: "processed_processor2Hello"
12:15:24 [INFO] Received: "processed_processor1Hello"
12:15:25 [INFO] Received: "processed_processor2Hello"
12:15:26 [INFO] Received: "processed_processor1Hello"
```

```
#!/bin/bash
mq2config="@CMAKE_SOURCE_DIR@/MQ_samp_proc_sink/options/MQ2_sampler_proc_sink.json"
```

```
SAMPLER="mq1-sampler"
SAMPLER+=" --id sampler1"
SAMPLER+=" --mq-config $mq2config"
xterm -geometry 80x23+0+0 -hold -e @CMAKE_BINARY_DIR@/bin/$SAMPLER &
```

```
PROC1="mq2-proc"
PROC1+=" --id processor1"
PROC1+=" --mq-config $mq2config"
xterm -geometry 80x23+0+300 -hold -e @CMAKE_BINARY_DIR@/bin/$PROC1 &
```

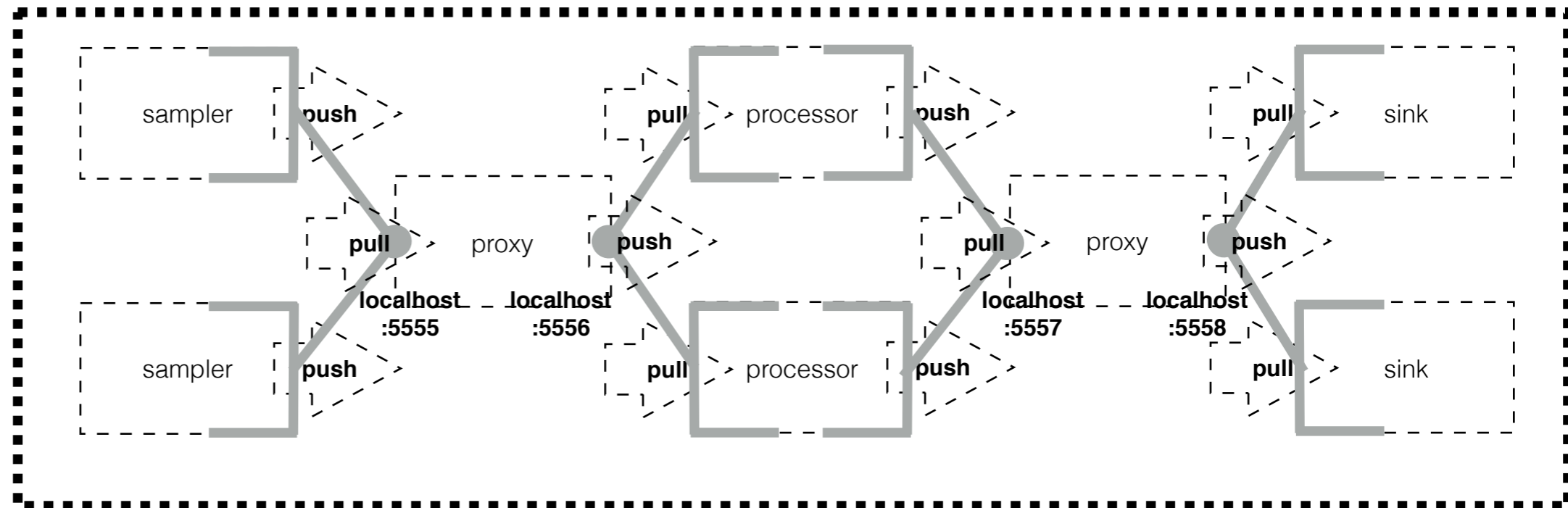
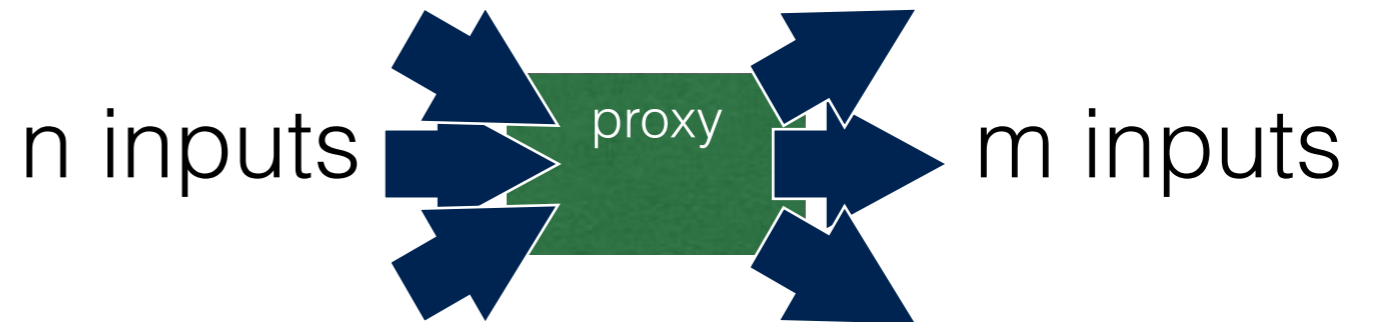
```
PROC2="mq2-proc"
PROC2+=" --id processor2"
PROC2+=" --mq-config $mq2config"
xterm -geometry 80x23+500+300 -hold -e @CMAKE_BINARY_DIR@/bin/$PROC2 &
```

```
SINK="mq1-sink"
SINK+=" --id sink1"
SINK+=" --mq-config $mq2config"
xterm -geometry 80x23+500+0 -hold -e @CMAKE_BINARY_DIR@/bin/$SINK &
```


Day 1. Proxy

workshop/build/FairRoot_dev-08.06.2017/bin/proxy

- use generic proxy device, able to connect nxm devices
- implement topology with 2 samplers, proxy, 2 processors, proxy, 2 sinks



```

mq1-sampler
[13:50:00] INFO: Running...
[13:50:00] INFO: Use keys to control the state machine:
[13:50:00] INFO: [h] help, [p] pause, [r] run, [s] stop, [c]
[13:50:00] INFO: next task, [d] next device, [q] end, [i] init task, [t]
[13:50:00] INFO: next device
[13:50:01] INFO: Sending "Sample1"
[13:50:02] INFO: Sending "Sample1"
[13:50:03] INFO: Sending "Sample1"
[13:50:04] INFO: Sending "Sample1"
[13:50:05] INFO: Sending "Sample1"
[13:50:06] INFO: Sending "Sample1"
[13:50:07] INFO: Sending "Sample1"
[13:50:08] INFO: Sending "Sample1"
[13:50:09] INFO: Sending "Sample1"
[13:50:10] INFO: Sending "Sample1"
[13:50:11] INFO: Sending "Sample1"
[13:50:12] INFO: Sending "Sample1"
[13:50:13] INFO: Sending "Sample1"
[13:50:14] INFO: Sending "Sample1"

mq2-proc
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)

mq1-sink
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"

mq2-proc
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)
[13:50:00] INFO: Sending processed_processor(Sampler1)

mq1-sink
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"
[13:50:00] INFO: Received: "processed_processor(Sampler1)"

[13:49:59] INFO: data in[0]: using default transcoder.
[13:49:59] INFO: Attached channel data-in[0] to topic/#555
5 (bind)
[13:49:59] DEBUG: Validating channel "data-out[0]"... WILD
[13:49:59] DEBUG: data-out[0]: using default transcoder.
[13:49:59] DEBUG: Attached channel data-out[0] to topic/#55
5 (bind)
[13:49:59] STATE: Entering SERVICE_READY state
[13:49:59] INFO: Test size (CPU): 0.90 ms
[13:49:59] INFO: Test size (Mem): 1.04 ms
[13:49:59] STATE: Entering INITIALIZING TASK state
[13:49:59] STATE: Entering READY state
[13:49:59] STATE: Entering RUNNING state
[13:49:59] INFO: SERVICE: Running...
[13:49:59] INFO: Use keys to control the state machine:
[13:49:59] INFO: [h] help, [p] pause, [r] run, [s] stop, [c]
[13:49:59] INFO: next task, [d] next device, [q] end, [i] init task, [t]
[13:49:59] INFO: next device
]
cp .../trino/mq_samp_proc_sink/options

```

Day 1. Proxy

