

Project 1

Simulation of proton microbunch events

Mu2e docdb-5225

Andrei Gaponenko

Mu2e Computing review 2015-03-05

Introduction

- ▶ Bob showed a list of background and accidental processes we need to simulate
- ▶ Mu2e dataset 3.6×10^{20} protons: brute force does not work \implies resampling, multi-stage job setup
- ▶ Design sims to have a workable trade-off between CPU and storage needs
- ▶ Manage $\mathcal{O}(10^6)$ jobs. Validation, bookkeeping. . .
- ▶ Three large simulation projects:
 - ▶ Signal and beam background in tracker+calo
 - ▶ Cosmic background (Ralf's talk)
 - ▶ CRV dead time (Yuri's talk)

This talk mostly summarizes our past experience. Ray will cover the upcoming improvements.

General workflow notes

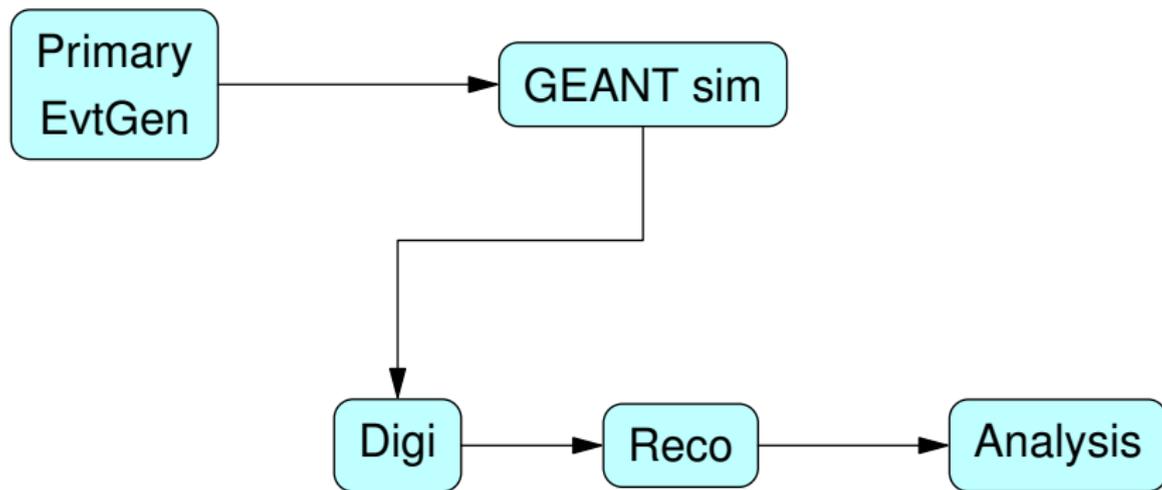
Consistency controls

- ▶ Use Mu2e Offline
- ▶ Generate and use common MC samples
- ▶ Follow a checklist to run production jobs
- ▶ Submit grid jobs from a dedicated account

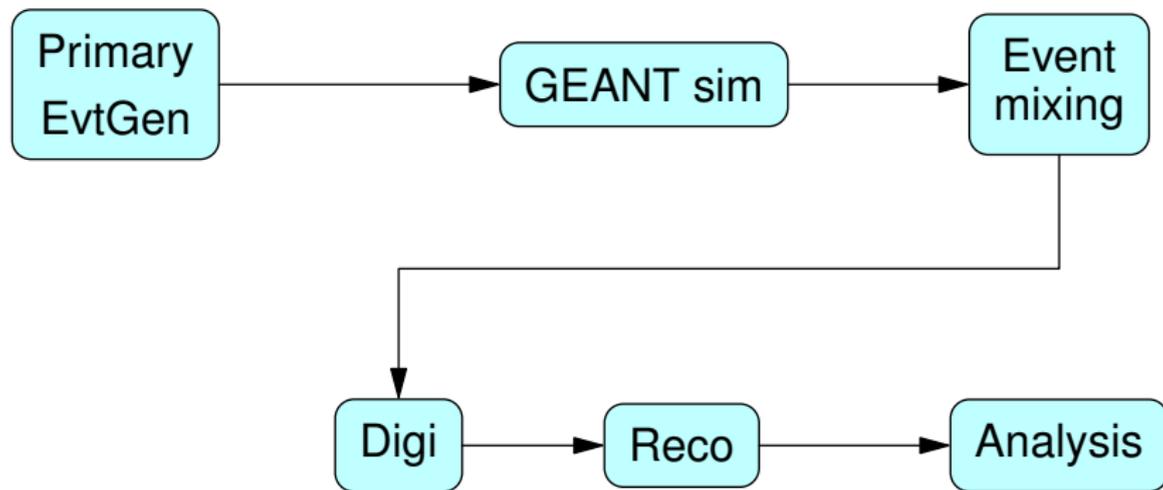
Reproducibility

- ▶ Code, geometry, fcl files are version controlled
- ▶ Production code builds are in a standard place
- ▶ Non VC files (e.g. a list of files for processing) are never modified, and are backed up
- ▶ Detect failed jobs and exclude from further processing
- ▶ Common MC samples are documented

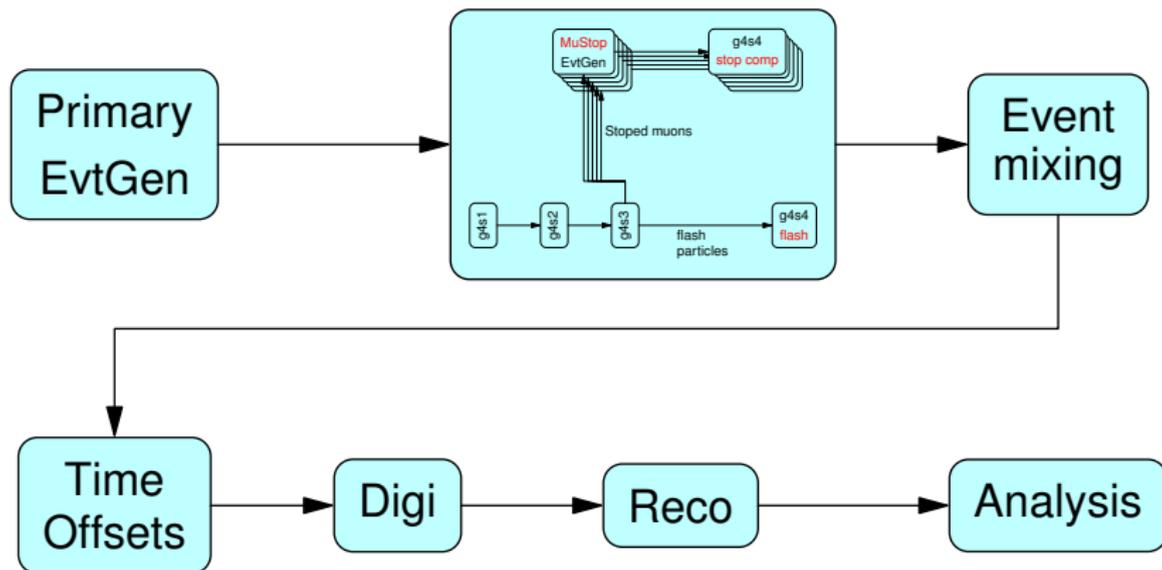
Stages of event processing



Stages of event processing with pile-up

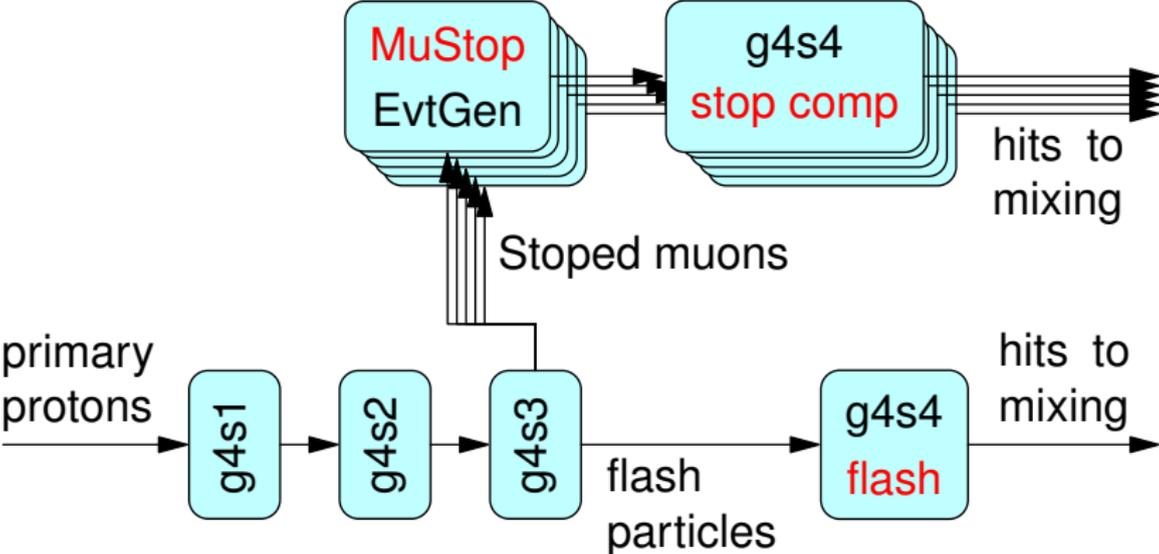


Stages of event processing in Mu2e



Time offsets due to proton pulse shape and muonic AI life time are postponed: better effective statistics for re-sampling, and use the same simulations for “in time” and “out of time” protons.

Mu2e simulation stages

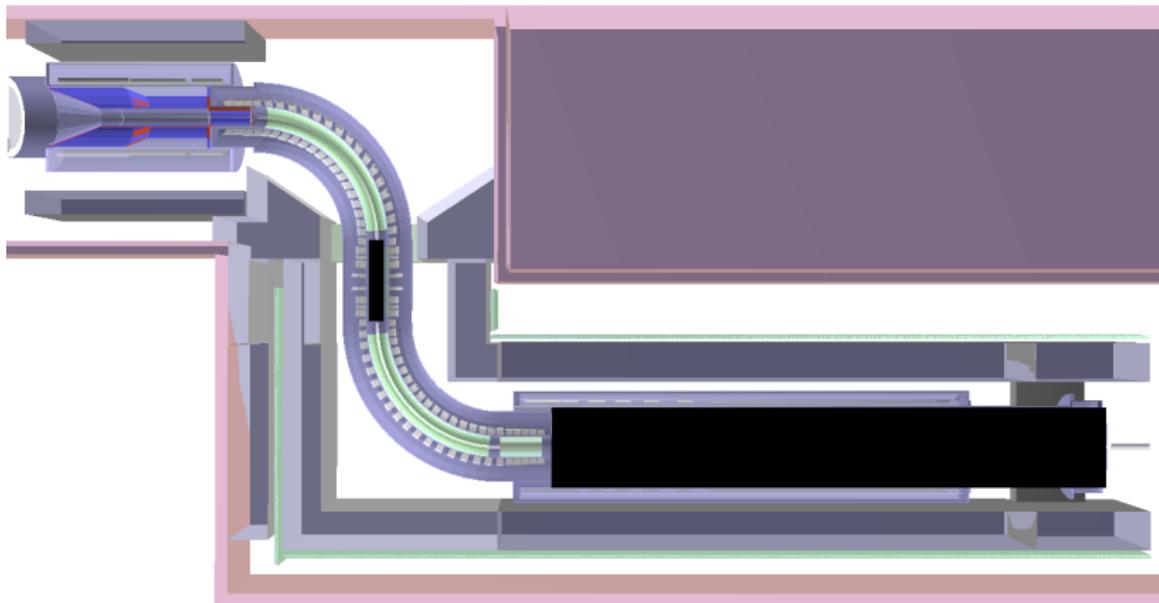


Example G4 Stage 1

Shoot protons on target, trace the secondaries.

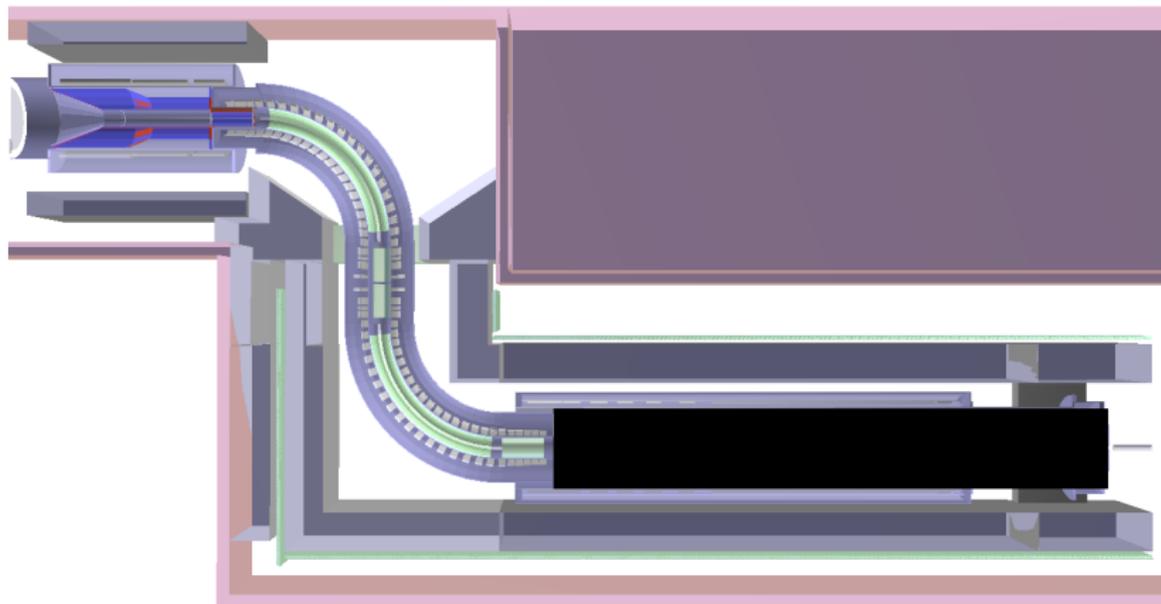
This is where most CPU time is spent.

Stop particles entering TS3 vacuum.



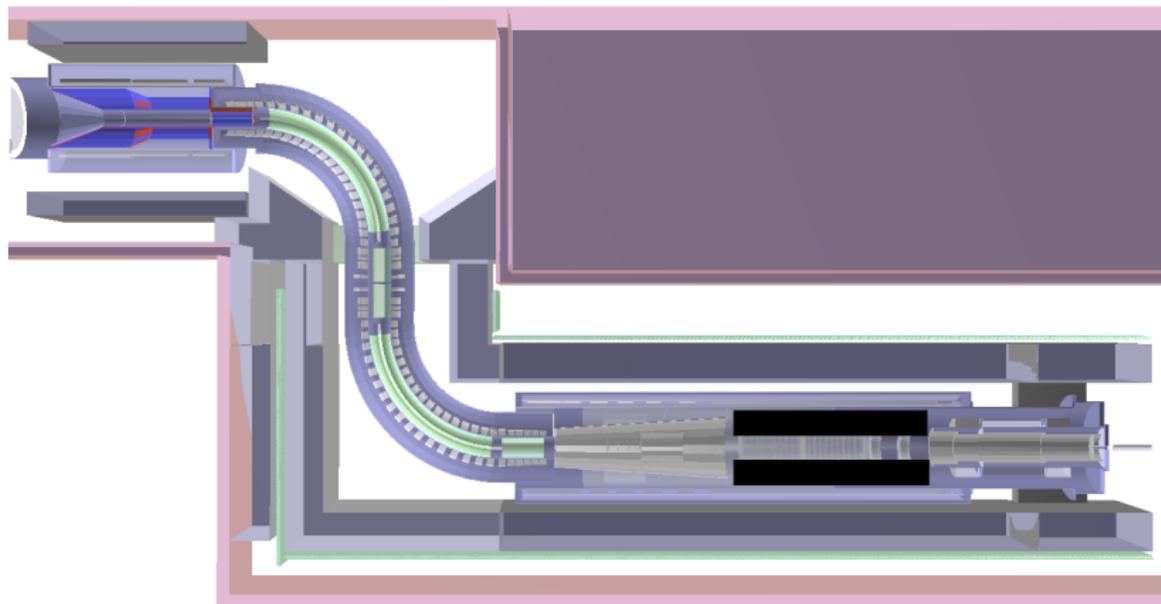
Example G4 Stage 2

Stop particles entering DS vacuum.



Example G4 Stage 3

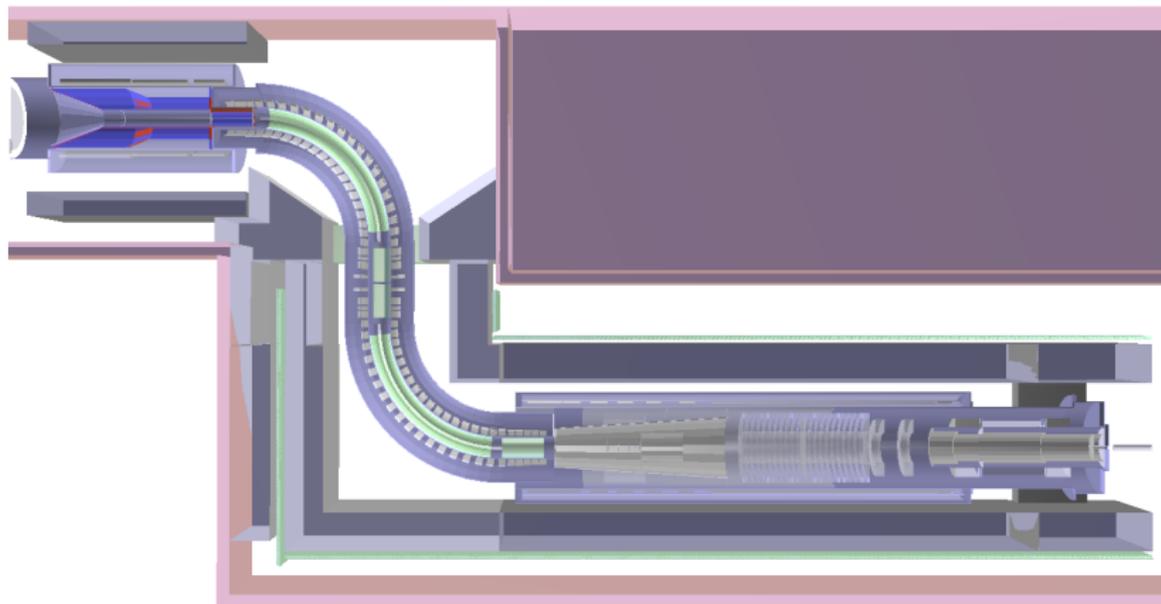
Stop particles entering detector “mother” volumes: “beam flash”
Write out stopped muons separately.
Filter out stopped muon daughters.



Example G4 Stage 4

The final stage: no opaque volumes.

Start with the stopped muons, or particles hitting det. mothers
Record hits in the tracker and calorimeter.



Multi-stage simulation benefits

- ▶ Allows to use different G4 cuts and/or physics lists to simulate different parts of the setup
- ▶ Easier to make changes to Mu2e configuration.
For example
 - ▶ s1 and s2 are not affected by changes inside the DS
 - ▶ Simulations for positive and negative muon beam polarity use the same s1
- ▶ Easier to correct mistakes.
For example:
 - ▶ Found a typo in G4 material definition
 - ▶ Fix and re-run: 500 CPU-hours instead of 2×10^6 CPU-hours

Multi-stage simulation comments

- ▶ Extensively use framework provided functionality to implement the complicated processing structure: filters, multiple trigger chains and multiple output streams.
- ▶ Several processing chains: “beam”, “pions”, “cosmic”, ...
- ▶ How much time does it take to simulate one event?
 - Which event?

| | |
|---|-------------------------------|
| <i>a single conversion electron</i> | $70 \times 10^{-3} \text{ s}$ |
| <i>a single microbunch event without re-sampling</i> | $1 \times 10^{+8} \text{ s}$ |

Passing event data across the stages

- ▶ Each stage writes out an art framework format file: position, momentum, time of a particle entering a “kill” volume
- ▶ Only particles that can eventually make hits in a detector are kept
- ▶ If a particle is saved, all of its parents and their “virtual detector” hits are preserved
- ▶ Subsequent stages **add to** particle MC history
- ▶ Stopped particle handling is different
 - ▶ Write out position and stop time as a ROOT Tree
 - ▶ This represents a PDF used by the next stage **event generator**

Passing metadata across the stages

- ▶ Need info on sample normalization at the analysis stage
- ▶ Can not just count event in analysis or use the nominal number of events in a sample because of the grid job loss and event filtering
- ▶ Mu2e approach: rely on atomic subruns (“luminosity blocks”)
 - ▶ Never split a subrun across different files
 - ▶ Production and analysis jobs must read complete files (no exit midway)
 - ▶ The event generator job records normalization info into subrun object
 - ▶ Analysis jobs integrate info from subruns they see
- ▶ art framework supports what we need

Event mixing

- ▶ G4 sims s1–s3 and “beam flash” s4:
generated event = 1 proton on target
- ▶ G4 sims s4 except beam flash:
generated event = 1 particle stopped inside the DS
- ▶ Mixing output:
mixed event = 1 microbunch of 31.5×10^6 POT
- ▶ Mixing functionality is provided by art + Mu2e specific code
- ▶ Common mixed sample: “analog” hits for background-only events. Signal or high momentum background tracks are overlaid on top by various analyses.

Mixing complete microbunch-events.

- ▶ 32-bit `cet::map_vector` key is too small: used a workaround
- ▶ Need 8 GB memory/process
- ▶ Demonstrated for tracker+calo, but not used in production
- ▶ Upcoming CRV and other studies may benefit from mixing complete events
 - ▶ Would like to have a 64 bit key.
 - ▶ Access 8GB RAM or better grid nodes. (Using them should be much easier now with the new `jobsub_client` based grid scripts.)
- ▶ But most studies did not need hits far away from the analysis time window, so . . .

Mixing: detector live window

- ▶ Pre-filter mixing input to discard early hits
 - ▶ Some “fuzz” around the cut, but there is a safe cut value
- ▶ Maximize input file size for better mixing.
 - ▶ 10 GB total inputs
 - ▶ 10 GB output file; everything fits on worker node disk
 - ▶ \implies **Mixing jobs are very I/O intensive**. Could submit only 100 at a time, or lose everything due to I/O locks and grid job time limits.
- ▶ Mixing worked OK for hits in the detector live window
- ▶ The “live window” mix is $\mathcal{O}(20)$ MB/event, so **all digitization and analysis jobs read large volumes of data**

Re-sampling is crucial to get sufficient statistics

Stopped muon resampling

- ▶ Randomized time and **direction** of ejected particles. A safe resampling factor for a single muon is

$$\begin{aligned} & (\Delta t_{\text{live gate}} / \Delta t_{\text{detector}}) \times (4\pi / \Delta\Omega_{\text{detector}}) \approx \\ & (1000 \text{ ns} / 50 \text{ ns}) \times (4\pi / (25 \times 10^{-6})^2) = 4 \times 10^{11} \end{aligned}$$

- ▶ Once the stopping target is reasonably populated with stopped muons, we can re-sample them “forever”

Re-sampling is crucial to get sufficient statistics

Beam flash resampling

- ▶ The only randomization is through the proton pulse shape time offset. A “safe” resampling factor for a single particle is $\Delta t_{\text{pulse width}}/\Delta t_{\text{detector}} \approx 250 \text{ ns}/50 \text{ ns} = 5$
- ▶ Hits from 2300 different beam flash parent protons are re-sampled independently: $5 \rightarrow 5^{2300}$: we are ok for “salt and pepper” hits
- ▶ However effects from correlated “beam flash” hits (e.g. beam electron tracks) are very hard to simulate: no large re-sampling factor.

Grid use

- ▶ Utilizing grid with opportunistic cycles was crucial
- ▶ Overall performance was good, got results on time
- ▶ **mu2egrid** is a Mu2e-specific set of scripts.
 - ▶ Uses `jobsub_client` to do the work
 - ▶ **Multiple job submission in a single command**: template fcl file is modified by individual worker processes to customize random seeds and input files for mixing.
 - ▶ **Validation step**: check a *positive* indication of job exit status 0. Jobs are sorted into “good”, “failed”, and “re-run” (duplicates).

Need an update to write to pnfs/SAM instead of bluearc.

Grid issues

- ▶ Jobs randomly killed for no reason. Saw this at a percent level. A cluster of 1000 8-hour jobs on a lightly loaded grid could take 3 days to complete: a few jobs are killed and restarted, more than once. Bulk of the cluster would be done quickly, but there was a long “tail” of “unlucky” jobs.
- ▶ Saw more than one instance of the same job \implies duplicate outputs.
- ▶ Mu2e production jobs were run from the mu2epro group account. **But grid submission required someone “donating” a personal proxy!** Impacts that user’s individual priority.
 - ▶ Supposed to be resolved by the newest `jobsub_client`