GRCon21
CHARLOTTE, NC

Maintainers' Update

Jeff Long, Josh Morman - co-maintainers

Introduction

Thanks to Marcus for 3 years of solid maintainership and continued leadership on all code aspects of the project

Change in maintainership early 2021

- Jeff Long - stable release maintainer (maint-* branches)
    - willcode4@gmail.com
    - github: willcode
- Josh Morman - dev branch maintainer (master/dev branch)
    - jmorman@gnuradio.org
    - github: mormj

Release Strategy

Maintenance Releases

- Aggressive backporting
- Minor changes in stable vs. frequent ports to new version
- Keeping branches consistent makes maintenance easier
- Quarterly cadence to provide more choices for downstreams

Major Releases

- Keep the master branch as a development branch
- When there are enough non-backportable features (API/dependency changes), make a major release (next is 3.10)

Maintenance Branches

Some great questions

- How many will we keep active?
- How long will we support maintenance?
- Concept of LTS branches? - will that be 3.8?

Renaming the master branch

Based on trends in industry and communities, the current `` `master` `` branch will soon seem antiquated

We also want to recognize that we allow more freedom in this branch, and use it primarily as a development branch

For this reason, we are proposing after 3.10 is released, rename to `` `dev` ``

- We already require maint-* branches to be stable
- Our main branch can be a bit more experimental to encourage code velocity
    - Note which features/fixes need more play before backporting

Process - master/dev

- Git source code control
- Github hosting, PR and issue management, CI infrastructure
- All changes submitted via PR (no direct commits)
- Linear history per branch, commits are rebased, not merged
- PR checks
  - DCO
  - Formatting using clang-format
  - Build/test on multiple targets
- Review by devs and maintainers, approval
- Process can take time due to schedules, need for experts, indecision
- PRs are applied to master first, backport completed when “safe”

Process - backporting

- Consider PRs that would be (1) useful and (2) cause no harm
- Apply “Backport-3.x” labels
- Author input on backporting encouraged
- PRs that are identical master <-> maint
  - Preferred if at all possible!
  - Backported by maintainers
  - Often this is easiest for everyone
- PRs with differences master <-> maint
  - May need significant author support
  - Author: point out the differences
- “ported-to-3.x” labels where backport PR has been created

What gets backported

- Current maint (maint-3.9)
    - Tracks master wherever possible
    - Generally NOT build system or dep changes
    - API must be backward compatible
    - Nothing that breaks existing flowgraphs
- Previous maint (maint-3.8)
    - Limited changes - stability and developer resources
    - Bug fixes
    - Occasional new blocks
    - Major performance improvements (bug fixes)
- GRC
    - More of an application, user facing
    - Keep branches as close as possible

Releases

- Source: Github releases, git clone -b v3.X.Y.Z
- Release numbering 3.X.Y.Z, 3.X.Y.0, 3.X.0.0, 4.0.0.0
- GNU Radio prioritizes API compatibility, many libs prioritize ABI
- PGP Keys - GNU Radio (Software Signing Key) <info@gnuradio.org>
  - With Github releases, also https://github.com/willcode/gnuradio-keys
  - Keyservers: keys.openpgp.org and pgp.mit.edu
  - Tags and releases are signed
- 3.8 releases with/without VOLK submodule included
- Example downstreams
  - Distros (Ubuntu, Debian, Fedora, ...)
  - Environments (Conda)
  - Containers (GNU Radio CI, various user created, ...)
  - PyBOMBS
  - Installers (Windows)

Maintenance is a Team Effort

We are trying to keep the PR queue *manageable* ... how can you help?

- Submit a PR, review a PR (or 2 or 3)
  - Reviews do not need to be a comprehensive line for line code (though someone with detailed knowledge should be scrutinizing the changes)
  - Test out PRs out in different scenarios, on different platforms
- Clean up draft PRs (or close them until they are ready again)
- Respond to comments in a timely manner
- Rebase and squash your commits (see CONTRIBUTING)
  - Commit messages and PR titles should include “module name:”
  - Include a short summary of the change
  - e.g. "digital: did some specific thing"

GR Releases - Year in Review

3.9.0.0
- pybind11 replaces SWIG
- C++ Modernization
- Updated dependencies
- boost::bind → lambdas
- more Boost removal
- python3 only

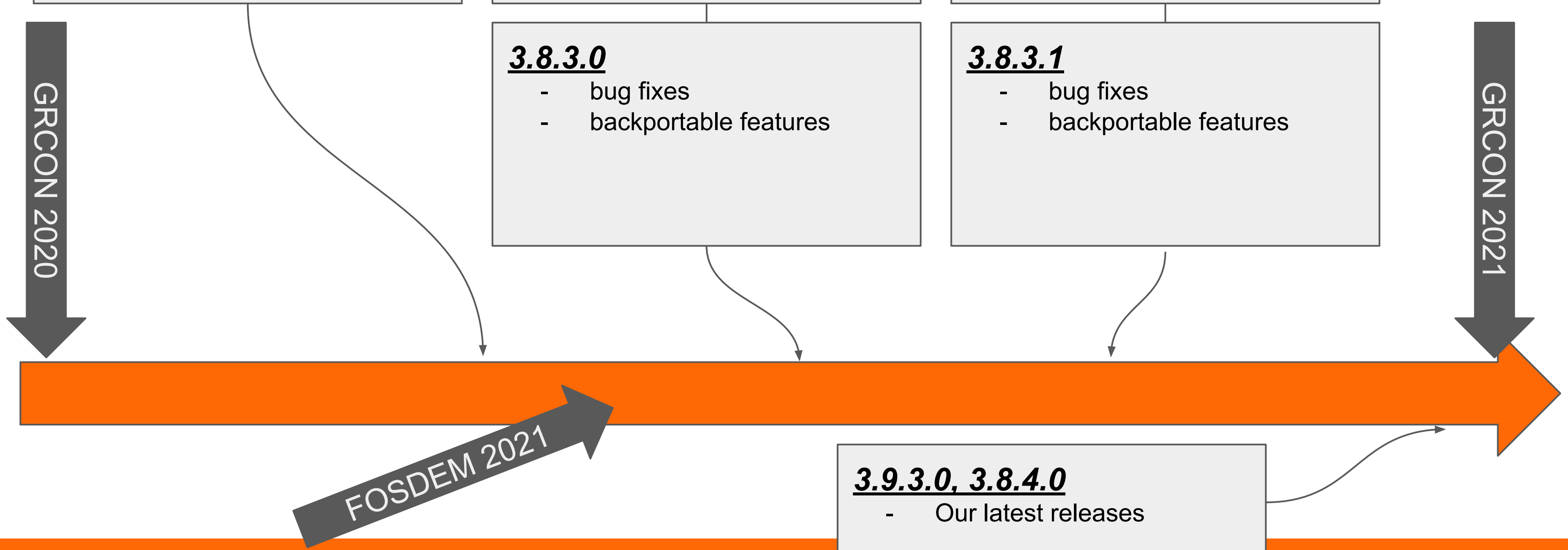
3.9.1.0
- better OOT support (cmake, modtool)
- grc bug fixes and fixed up features
- gr-uhd fixes

3.9.2.0
- gr-soapy!!! - in-tree access to LOTS of hardware
- grc parameter expressions
- bug fixes

3.8.3.0
- bug fixes
- backportable features

3.8.3.1
- bug fixes
- backportable features

3.9.3.0, 3.8.4.0
- Our latest releases



Upcoming Release 3.9.3.0

Why should you migrate to 3.9??

Most new features and fixes can be found on 3.9, only a subset make it back to 3.8

PyBind11 is easier to use and maintain AFTER the initial learning curve
https://youtu.be/7Aj4VHXTh-Y

Better supported, as developers working on 3.9 and newer version

Upcoming Release 3.8.4.0

Why would you stay on 3.8??

Developers

- Initial learning curve for PyBind11

Users

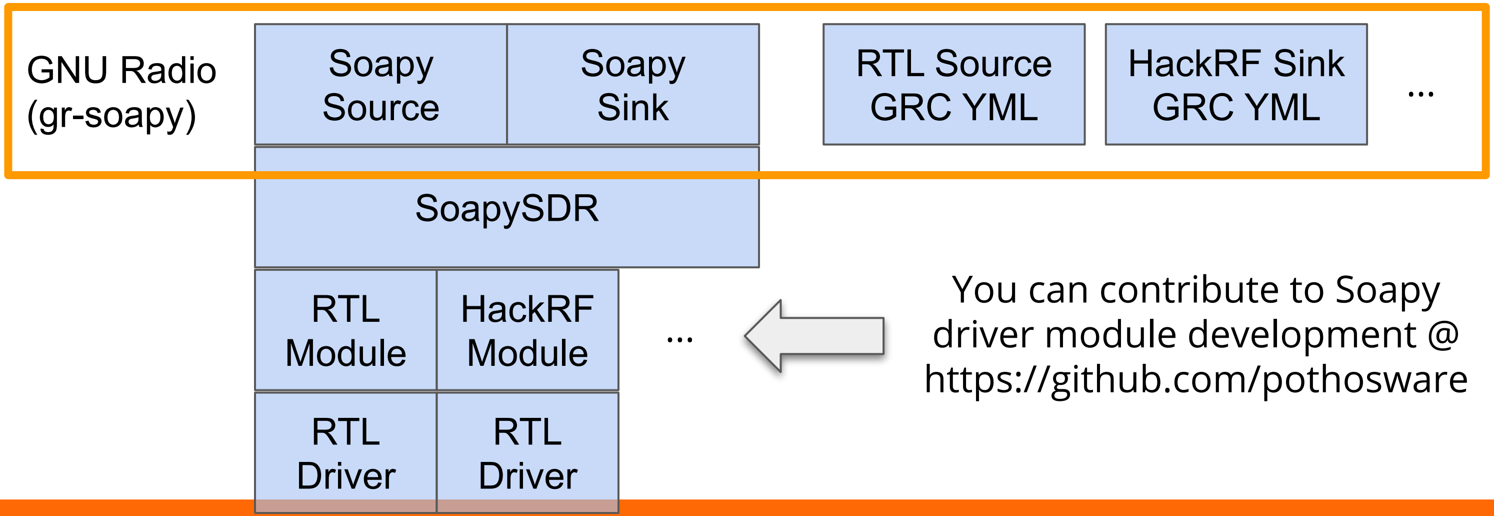
- OOTs may not have been ported to 3.9

gr-soapy

- SoapySDR is a driver framework, maintained by the Pothos
- gr-soapy
  - Original version by LibreSpace
  - Ported to 3.9, refactored, upgraded to wrap most recent SoapySDR
  - Does not contain driver-specific code
- Why add in-tree (was previously an OOT module)?
  - Decouple GNU Radio and driver development - modules are runtime loadable objects
  - Rich API supports a wide variety of hardware
- What about gr-osmosdr?
  - gr-osmosdr lives on, has many users
  - Two-way dependency delays availability when GNU Radio API changes
  - Contains driver-specific code, so best as an OOT module

gr-soapy: architecture

- gr-soapy - in-tree adaptor layer to make SoapySDR source/sink blocks
- SoapySDR - API, framework library
- Module - hardware-specific support for SoapySDR
- Driver - provided by “vendor”
- GRC YAML files - hardware-specific (and generic) instructions for GRC



You can contribute to Soapy driver module development @ https://github.com/pothosware

Major Releases

3.10 and near term future releases should not as big of a jump as recent major releases (3.7 → 3.8, 3.8 → 3.9)

OOTs *should* be able to indicate `find_package(gnuradio 3.9)` and work for both 3.9 and 3.10

OOTs *may break* because API is allowed to change, but there should not be fundamental changes that cause OOTs to have to go through major reconstruction

Blocks moving around, removed parameters, other various API changes

Upcoming 3.10 Release

gr-pdu

- Upstreamed from **Sandia National Labs (Jacob Gilbert)**
- Tools for manipulation of PDU objects
- See talk tomorrow

gr-iio

- Upstreamed from **Analog Devices**
- Source/Sinks for iio based devices
- See talk later this week

gr-sigmf - plan to upstream, 3.10 TBD

Upcoming 3.10 Release

Revamped Logging Infrastructure (**Marcus Müller**)

- "Good Riddance Log4CPP"
- Persistently difficult dependency to maintain and use
- spdlog provides much nicer and more modern logging facilities
- libfmt for boost::format string formatting replacement

Accelerator Device Support

- Upstreamed from **David Sorber (BlackLynx, Inc - DARPA SDR 4.0)**
- Streamlined data movement
- Device compatible buffer structure (single mapped)
- Talk later this week

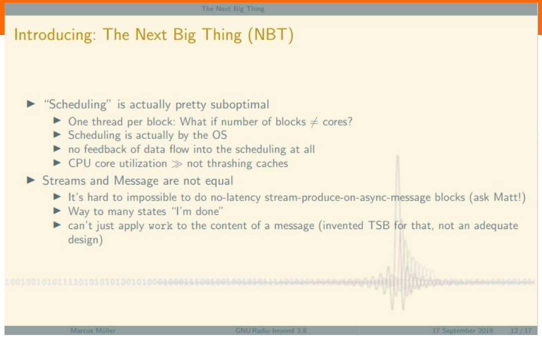
Maintainers Wishlist

The non code feature list of items we would like to improve upon
https://github.com/gnuradio/gnuradio/projects/9

- Automated Packaging
  - Is the ppa maintainable - if so, need better tools and a champion
  - Is conda the answer to get latest and greatest?
- Better Test Coverage
  - All grc examples
  - Flowgraphs with real data
  - QA test flakiness / non-determinism (e.g., random seeds)
- Native Windows Builds
  - Would be nice to double click on gnuradio.msi
  - Does radioconda supercede this?

GR 4.0

GR 4.0 - it begins now



Introducing: The Next Big Thing (NBT)

- "Scheduling" is actually pretty suboptimal
  - One thread per block: What if number of blocks $\neq$ cores?
  - Scheduling is actually by the OS
  - no feedback of data flow into the scheduling at all
  - CPU core utilization $\gg$ not thrashing caches
- Streams and Message are not equal
  - It's hard to impossible to do no-latency stream-produce-on-async-message blocks (ask Matt!)
  - Way to many states "I'm done"
  - can't just apply `work` to the content of a message (invented TSB for that, not an adequate design)

Why are we talking GR 4.0 when we haven't even released 3.10?

Take a look back to GRCON 2019: GNU Radio Beyond 3.8 - A Technical Outlook

- Marcus presented the "Next Big Thing"
  - Multiple blocks per worker - performance gains
  - Single Actor Model
    - Make streams and messages handled consistently
  - This is the work in progress for GR 4.0 - though we have expanded this vision
  - We *could* do some of these changes for GR 3.10, in fact "Custom Buffers" are being upstreamed as a feature
  - But, much of the vision requires a restructuring of GNU Radio that is more sustainable
    - Approaching as a separate effort

Vision for GNU Radio 4.0

Modular CPU Runtime

- Scheduler as plugin
- Application-specific schedulers

Heterogeneous Architectures

- Seamless integration of accelerators (e.g., FPGAs, GPUs, DSPs, SoCs)

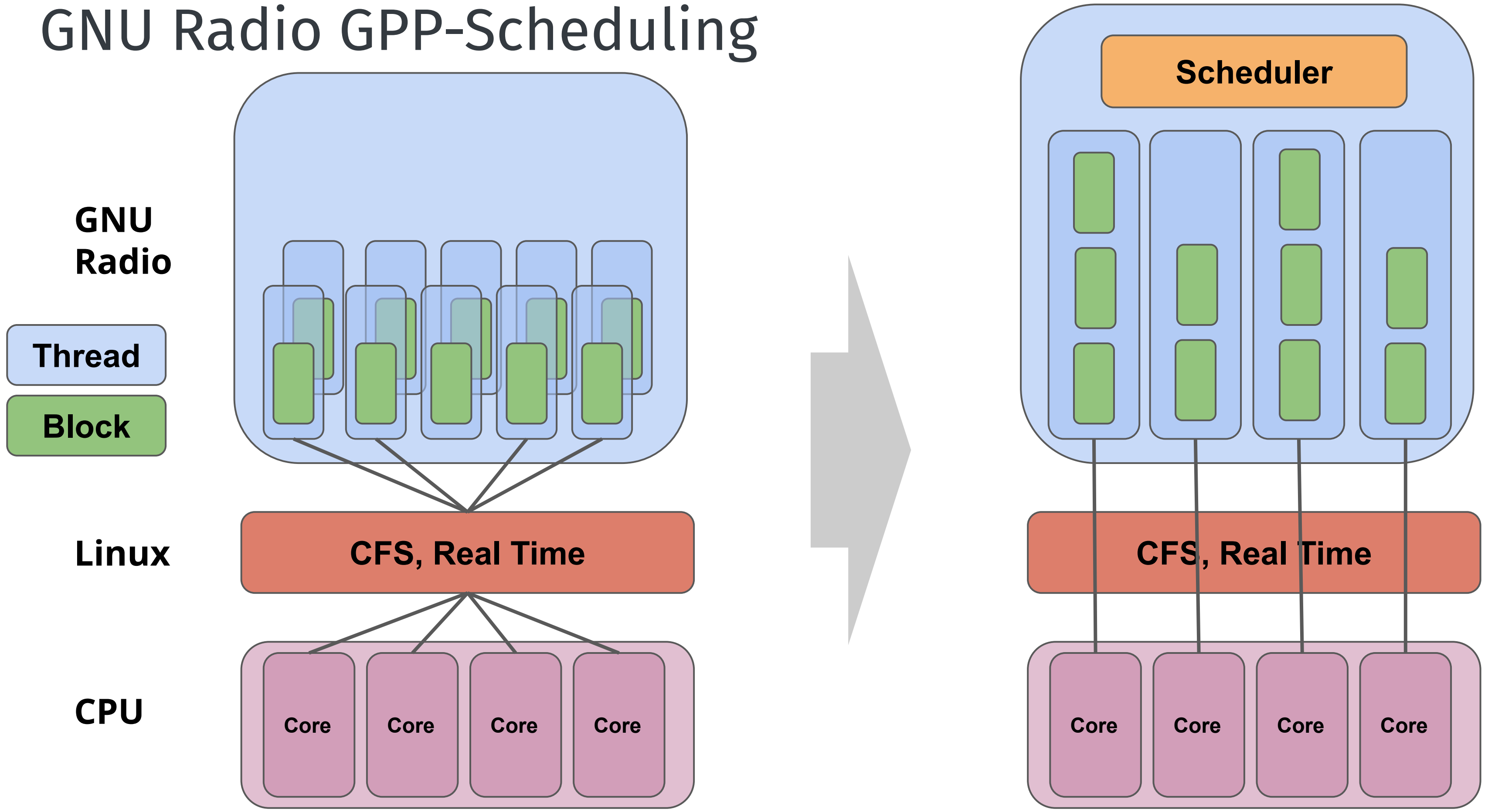
Distributed DSP

- Setup and manage flowgraphs that span multiple nodes

$\underbrace{\hspace{10em}}$

Straightforward implementation of (distributed) SDR systems that make efficient use of the platform and its accelerators

GNU Radio GPP-Scheduling



What is "newsched"

https://github.com/gnuradio/newsched

Project started at the pre-FOSDEM 2020 hackfest at the ESA, Noordwijk, Netherlands

Original Goal: A clean-slate approach to write a GNU Radio runtime that works for humans. -- get back to the basic principles

Since then

- our vision and goals have broadened
- the implementation is taking shape
- **newsched aims to be the basis for a GR 4.0 runtime**

Side Benefits

Beyond the main vision goals ...

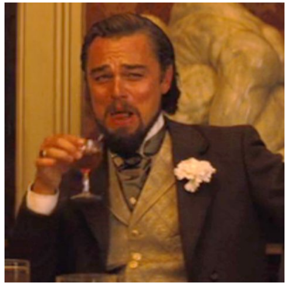
If we're "starting from scratch", what are other fundamental issues we'd like to overcome

- Simplified APIs (cut through some of the bulk of GR)
- A Better Developer Experience
  - Streamline the steps to "insert signal processing here"
  - More people using GR → More active development → More DSP blocks and features
- Callable work() function - accessible block kernels
- Natively Thread Safe
- Replace PMTs with something better

A Better Developer Experience

Case Study:

- Use gr_modtool to create a block
- Add a parameter to the make function
  - Update myblock.h
  - Update constructor declaration in myblock_impl.h
  - Add private variable in myblock_impl.h
  - Update make function, call to constructor and constructor in myblock_impl.cc
  - Update GRC file
  - Update the python bindings
  - Add setters/getters/callbacks
  - (Update msg port setters)
  - (Update RPC interface)
  - Do this for float, complex, int ...
  - Don't forget the docs!!!



🤯

How much of this could be centralized / automated with a new API / workflow??

GR 4.0 - Getting Involved

1. Join GR 4.0 Workshop (Thursday) to go into more detail about current status and future plans of newsched
2. Join Scheduler Working Group (next slide)
3. Try out the codebase
    a. Implement some blocks, make a flowgraph
    b. Relay your experience
    c. Make improvements
4. Get involved in development
    a. Port over a block or module from GNU Radio
    b. Review PRs
    c. File Issues

Scheduler Working Group

Following the breakout session from GRCON20, we have been meeting periodically - ~1x/month - times and meeting links shared via:

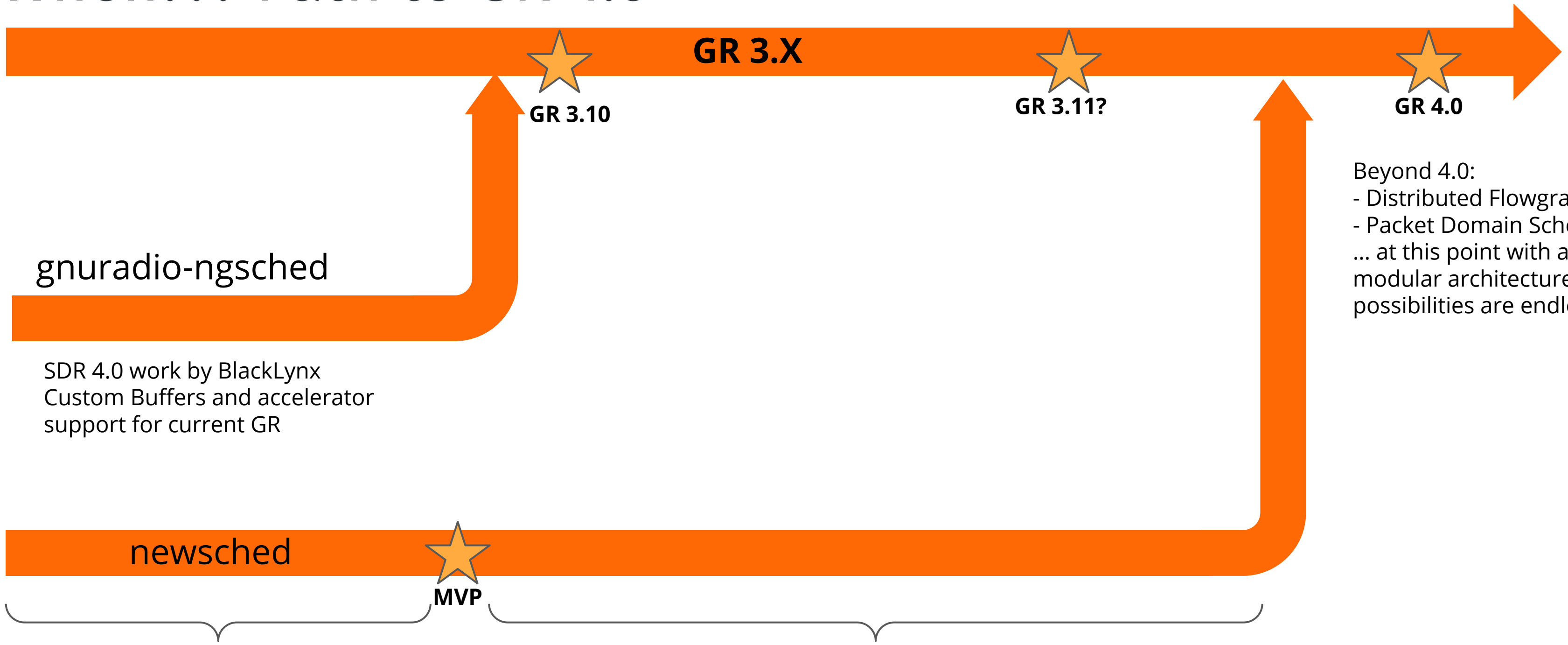
Chat room:
https://chat.gnuradio.org/#/room/#scheduler:gnuradio.org

Mailing List:
https://groups.io/g/gnuradio-scheduler

Some Topics Covered Thus Far:

- newsched status
- Custom Buffers
- Domains
- Hierarchical Scheduling
- Scheduling Paradigms
- Blocking I/O
- OpenCPI alignment
- Benchmarking
- SDR 4.0 Design Review
- PMTs
- Message Port interfaces
- GPU domain scheduling
- GRC Workflow Integration
- ...

When??? Path to GR 4.0



Beyond 4.0:
- Distributed Flowgraphs
- Packet Domain Scheduling
... at this point with a modular architecture the possibilities are endless

SDR 4.0 work by BlackLynx
Custom Buffers and accelerator support for current GR

Outstanding items for MVP:
- ?
- ...

Fill in the gap between what works in the MVP and GR 3.x