Design Considerations for Radio in CORSIKA8

Presented at CORSIKA Cosmic Ray Workshop

Remy L. Prechelt

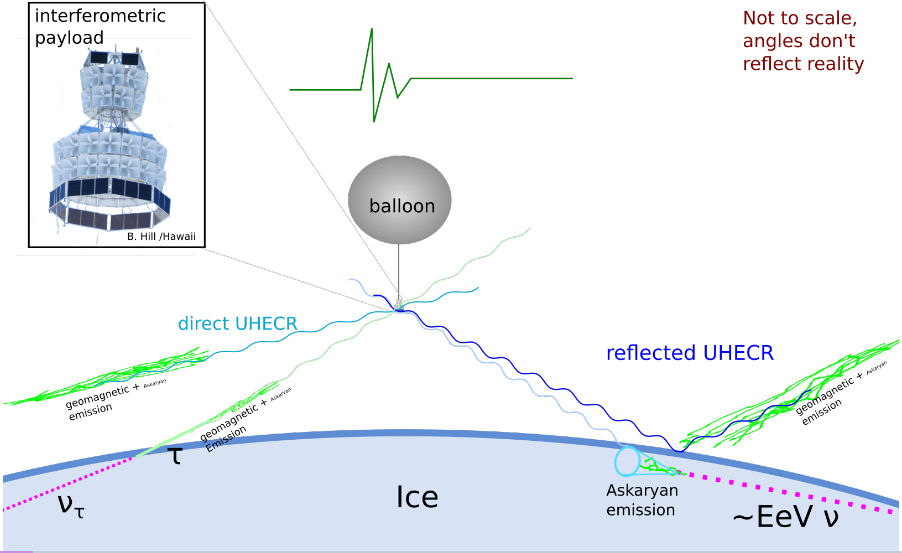
June 24, 2020

The “Now”

Current state of radio emission modeling for particle showers:

- Multiple formalisms (ZHS=>ZHAireS, Endpoint=>CoREAS) across multiple tools (Aires & CORSIKA7).
- *Typically* only supports downgoing showers for standard ground-based detectors.
- Hard to *modify and extend* for experiments that don’t perfectly fit the design assumptions. Collaborations may have to *hack* or *patch* modifications that are fragile and bug-prone!
- Not designed for simulating showers in other media (ice, regolith) and do not support custom environments (i.e. multiple media)

Example of Issues of Radio Simulation Ecosystem (ANITA/PUEO)



The "Future"

Goals for `CORSIKA8`:

1. We need to design the radio emission module to take full advantage of the *flexibility* of `CORSIKA8` so we can simulate a much wider variety of geometries and experiments.
2. We need to support both formalisms (ZHS & Endpoint) and a variety of heterogenous hardware (SIMD, MT CPU, MPI, GPU)
3. Standard steering and output file formats across all implementations and backends.
4. We need to make it easy for collaborations to *modify and extend* the radio module to support the needs of their particular experiment while maximizing code reuse.

This talk presents some of the early design discussions and interface requirements for feedback from the broader radio community. Please interrupt with questions!

Automatic Improvements & Status

The current C8 framework *already* provides clear advantages over existing simulations:

- The geometry & environment architecture can support arbitrary shower geometries (not restricted to downgoing air showers like C7 / Aires).
- The shower physics is being developed to support a wider range of media (in-ice showers!) and showers developing across multiple media (air-to-ice showers, etc.).

However, there are still several blocking elements before C8 can start to be used:

1. No EM interaction model.
2. No magnetic field modeling.
3. No radio emission implementation (← this presentation)

The goal is to develop the radio emission in *parallel* with the electromagnetic interaction model but *hadronic showers* could be run as soon as the radio implementation is ready.

Extensibility

While `C8` will provide *at least* two standard reference implementations for radio emission (ZHS and Endpoint), many experiments will need to develop custom variants to extend the features of `C8`.

1. ANITA must model the reflection of the radio emission off a rough ice surface and additional propagation upwards to the detector.
2. In-ice detectors that need to model the effects of refraction during propagation (talk by C. Glaser).
3. Particle showers penetrating from air into ice (talk by K. de Vries)

`C8` will be designed to make implementing these variants *as easy as possible* and maximize code reuse across the different implementations.

Status

A prototype CPU-based ZHS implementation is currently in development - there are still some non-radio tasks that must be completed before it can be tested but it is underway!

The discussions in the next few slides are based on this prototype.

The environment architecture in `C8` is modular so (automatically) support:

- Custom medium density models and geometries.
- Dependent (or independent) refractive index models (tabulated or analytic) (coming soon).
- Uniform, IGRF/WMM, or custom magnetic field models (coming soon).

Prototype Radio Process Architecture

- The prototype radio interface is composed of two compile-time subcomponents that can be independently swapped out:
  1. A specific *radio implementation* (Endpoint+CPU, ZHS+GPU, etc.)
  2. An antenna implementation (details later)
- This decouples simulation-wide parameters from antenna-specific parameters and allows for easily replacing the hardware backend of a particular simulation.

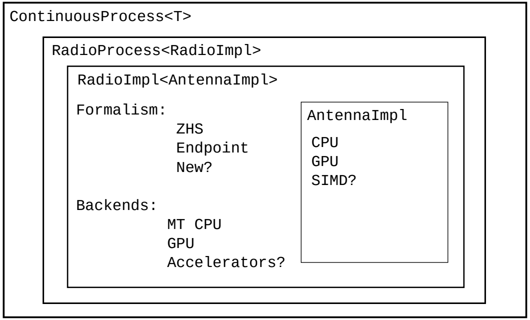


Figure 1: Compile-time polymorphic (CRTP) `RadioProcess<>` hierarchy.

Radio Implementation

- The `RadioEmission<RadioImpl>` interface implements common radio functionality:
  - Loading text-based antenna config files.
  - Processing and saving (HDF5?) output files (see T. Huege talk).
  - Common utility methods.
- Concrete implementations of the *actual radio formalism* must (currently) provide:
  1. `Emit(Particle&, Track const&)` or `Emit(Particle&, Track const&, Antenna&)` that calculates, propagates, and stores the radio emission from a given particle along a given track into each antenna.
  2. `MaxStepLength(Particle const&, Track const&)` to control the next allowable step length of the particle.
- Different concrete implementations (time or frequency domain) can be optimized and provided for the different backends (CPU, SIMD, MPI, GPU etc.) as instantiated by the user as needed.

Antenna Implementation

Concrete `Antenna<>` instances manage their own properties and waveforms. This allows for a variety of interesting optimizations:

- Optimizing waveform duration and start time for each antenna (memory usage optimizations)
- Different sample-rate/bandwidths for each antenna (reducing cache/memory usage where not needed)
- Custom orientation of the antenna polarization axes (can be placed to align with a particular experiment.)
- Incorporating experiment detector responses into CORSIKA simulation (using a custom `Antenna<>` instance).
- Others??

Most experiments will be served by the optimized implementations provided by `CORSIKA8`.

Parallelization

By decoupling the simulation formalism from the antenna implementation, lots of opportunities for parallelization are available:

- SIMD-optimized radio formalism and waveform assignment.
- *Lock-free* multi-{thread,process} parallelization over individual antennas.
- … and more!

Python Analysis Suite

- CORSIKA7+CoREAS provides a useful suite of Python tools for loading and processing CoREAS output.
- `C8` should provide a similar package to perform common analysis tasks from Python (loading output, coordinate transformations, filtering, etc.)

Join the Discussion

- There is a merge request open specifically for discussion regarding the design of the radio interface (here)
- If you are interested in contributing at this early stage, please join the discussion or reach out to me (`prechelt@hawaii.edu`)

Hopefully there's plenty of time for questions and comments!

What do you need from `CORSIKA8` radio?