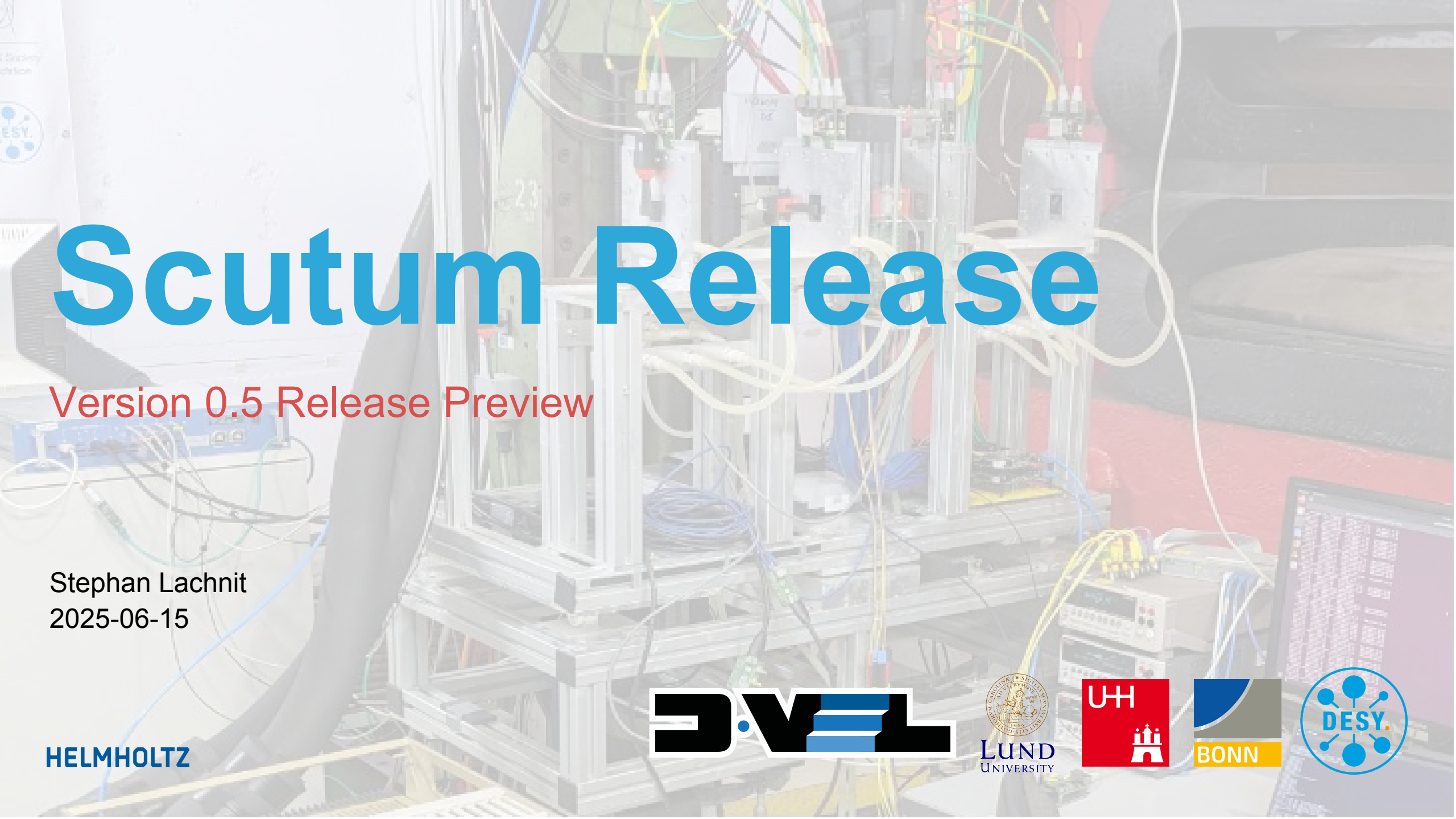


Scutum Release

Version 0.5 Release Preview

Stephan Lachnit
2025-06-15

HELMHOLTZ

Release Summary

Too many things for one slide

- **CHP roles for fine-grained heartbeating control**
- **Network discovery via multicast**
- **New FlightRecorder satellite for logging**
- **New LeCroy satellite for LeCroy Waverunner scopes**
- Heartbeating fixes in Python
- Error handling fixes in C++
- CLI Rework for easier network selection
- Documentation improvements
- ...

CHP Roles

Importance idea but fully autonomous

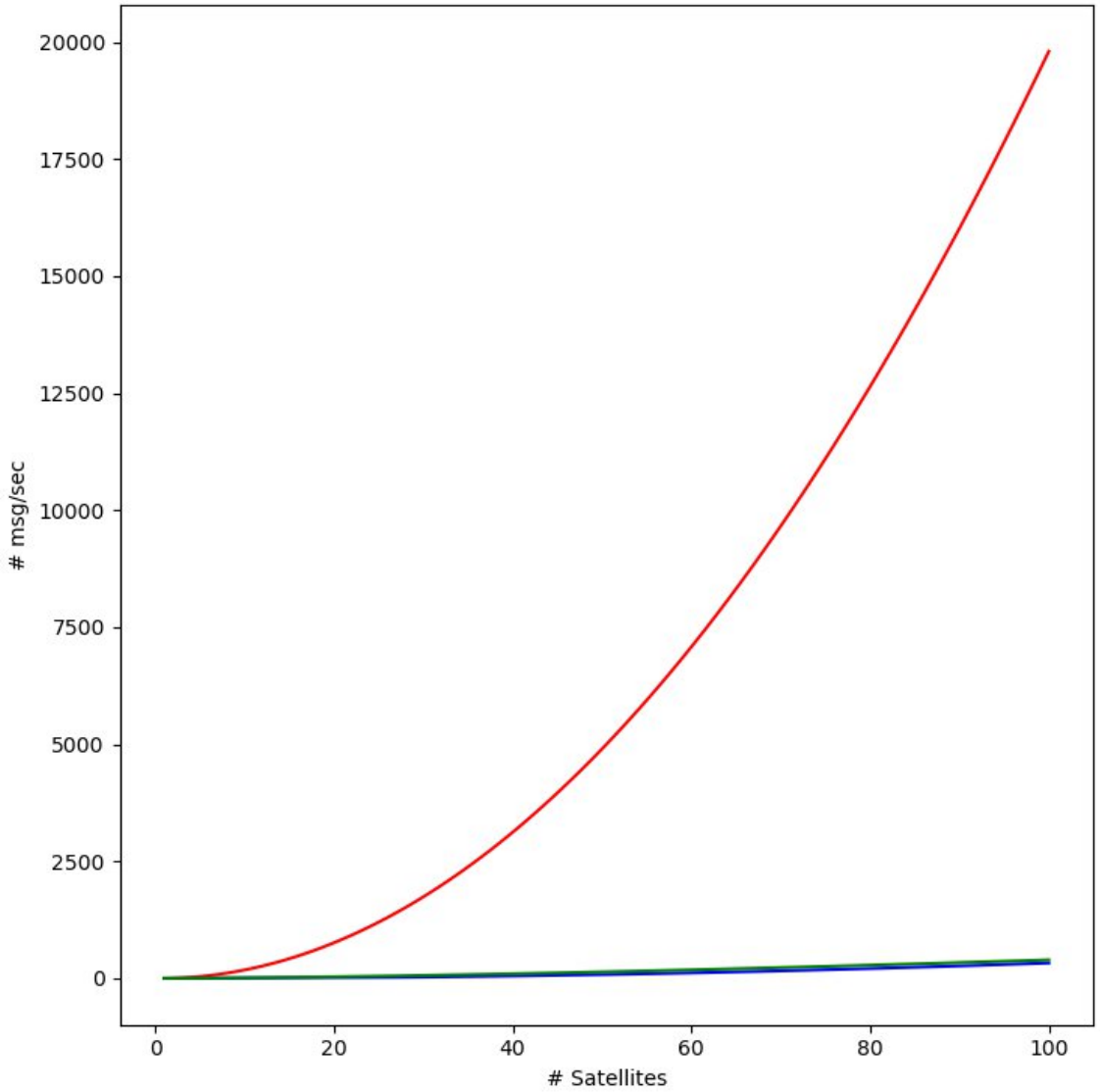
- Problem:
  - Any satellite disappearing during a run causes an inerrupt
  - However some satellite are not important enough (e.g. a logger or temperature sensor)
- Idea:
  - Each satellite has a role determining how other satellites should react if it disappears
  - Should be autonomous => heartbeating is the natural place to implement this
  - Three flags: `TRIGGER_INTERRUPT`, `DENY_DEPARTURE`, `MARK_DEGRADED`
  - Can be set by selecting one of the four roles: `NONE`, `TRANSIENT`, `DYNAMIC` and `ESSENTIAL`
  - Default role is `DYNAMIC` (`TRIGGER_INTERRUPT` and `MARK_DEGRADED`)

MR!713

CHP Congestion Control

Avoid flooding the network with heartbeats

- Problem:
  - The more satellite, the more heartbeat messages per second (red curve)
  - Low interval floods the network when having many satellites
  - Large interval is unnecessary when having just a few satellites
- Solution:
  - Scale the heartbeat intervall according to the number of satellites (blue/green curve)



MR!739

Multicasting in CHIRP

Making network discovery more robust

- Currently network discovery uses UDP broadcasts
- Problem:
  - Some firewalls and routers block UDP broadcasts
  - Doesn’t work (nicely) on MacOS
- Solution:
  - Switch to UDP multicasts
- Many improvements along the way:
  - Simpler selection of network interfaces (by name instead of broadcast address)
  - Proper handling of dead satellite being restarted

FlightRecorder

File logging made easy

- Can be used to store log messages to a file
- Different methods:
  - Single file
  - Rotating log files
  - Daily log files
  - Log file per run
- Usage hint: set `_role` to `NONE`

Flight Recorder Satellite

Name	Flight Recorder
Description	Satellite to record log messages from a Constellation
Category	Monitoring
Language	C++

Description

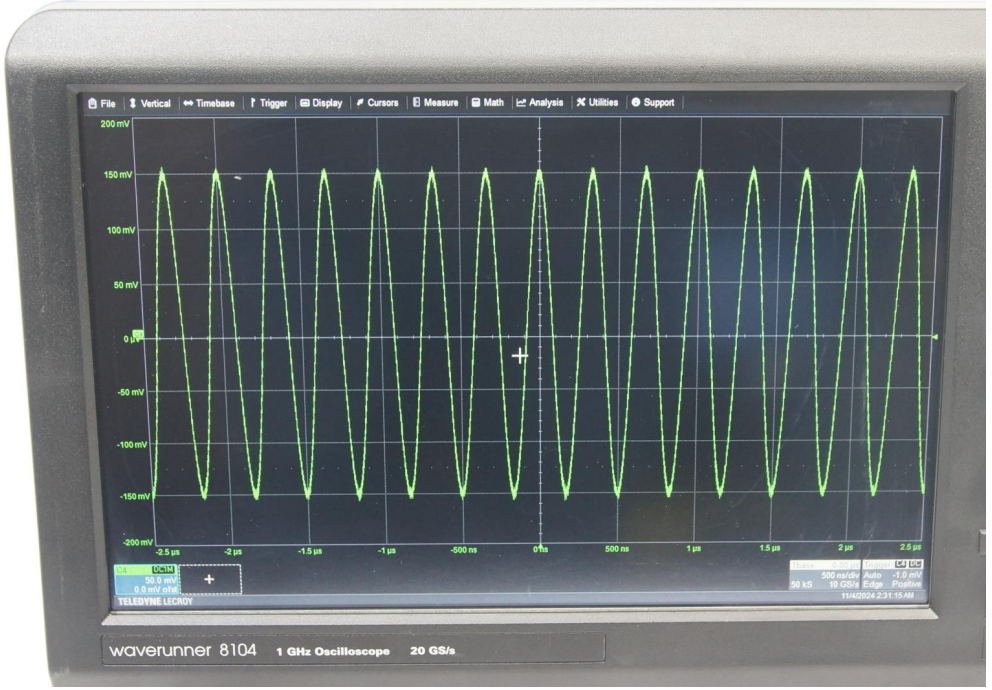
This satellite subscribes to Constellation log messages of a configurable level, receives them and records them to storage. Several storage options are available and can be selected via the `method` configuration parameter:

- `FILE` represents the simplest storage, all logs are recorded into a single file provided via the `file_path` parameter.
- `ROTATE` allows to use multiple, rotating log files. Every time the current log file reaches the size configured via the `rotate_filesize` parameter, a new file is started. After the maximum number of files set by `rotate_max_files` is reached, the oldest log file is overwritten. This can be useful to keep only a certain amount of history present.
- `DAILY` will create a new log file every 24h at the time defined with the `daily_switch` parameter. This method is particularly useful for very long-running Constellations which require preservation of the log history.
- `RUN` will switch to a new log file whenever a new run is started, i.e. when this satellite received the `start` command. This can be especially helpful when many runs are recorded and an easy assignment of logs is required.

LeCroy

First scope satellite

- Satellite to store waveforms from LeCroy scopes
- Sends waveforms via CDTP
- Contributed by Laurent Forthomme from CERN



LeCroy Satellite

Name	LeCroy
Description	Satellite controlling a LeCroy oscilloscope using the LeCrunch library
Language	Python

Description

This satellite uses the LeCrunch3 library by Nicola Minafra, based on LeCrunch2 and LeCrunch, to control and fetch each channel waveform from a `LeCroy Waverunner` series scope. Using a TCP connection to the scope's `Waverunner` software, it provides a simplified method to start a single- or sequenced-waveform acquisition and transmits it through the CDTP to Constellation receivers.

Requirements

This satellite requires the `[lecroy]` component, which can be installed with:

PyPI | Source

```
pip install ConstellationDAQ[lecroy]
```

MR!407

Conditional Transitions

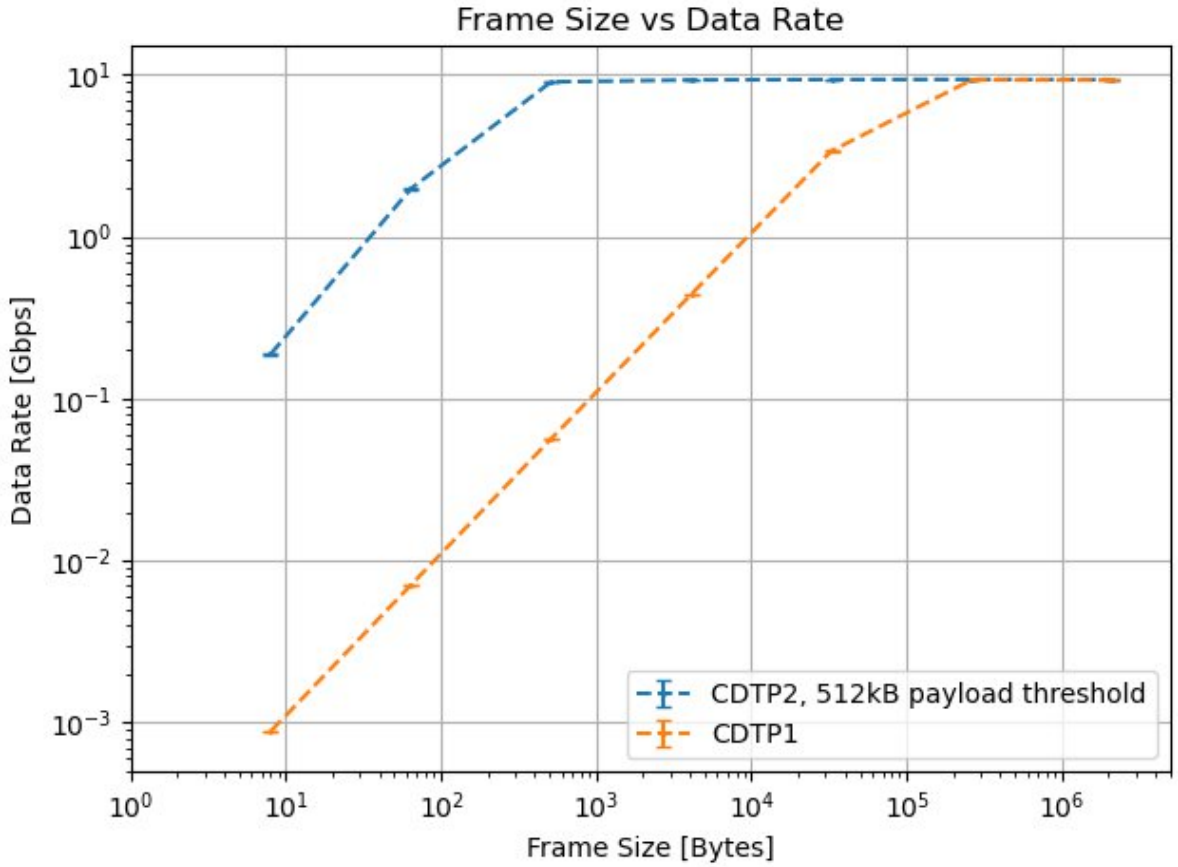
Waiting for other satellites

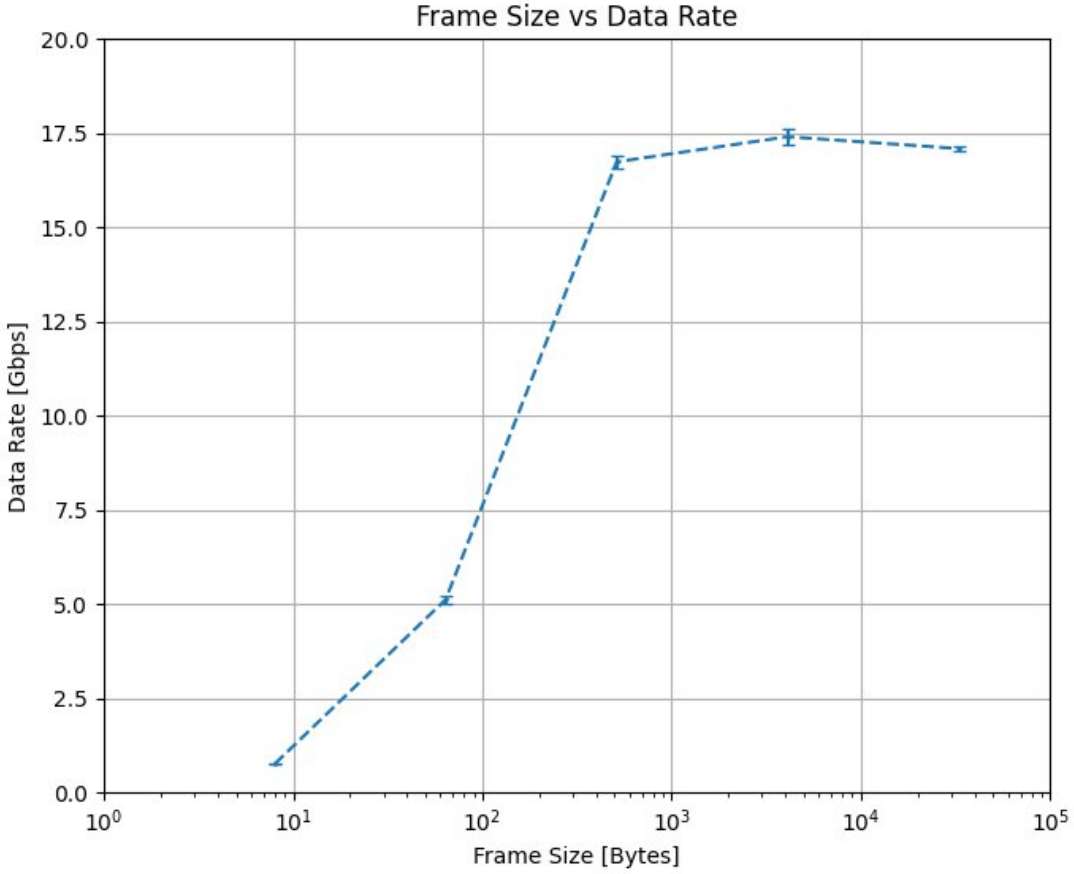
- Problem:
  - Some transitions need to be executed in a specific order
    - For example, a TLU distributing a clock might need to initialized before other detectors
  - Clicking buttons in a GUI in a specific order is bad UX
  - There is not central control instance => solution should be autonomous
- Solution:
  - Each satellite can be given a list of satellites to wait for before executing the transitions
    - No logic required on the controlling side
    - Fully autonomous

MR!659

Preview for v0.6: CDTP2

Transferring data at >10Gbps





Note: breaking change for most data transmitting / receiving satellits