Constellation

First Concepts for a SCADA System

Simon Spannagel, DESY

1st EDDA Meeting
20/09/2023

SCADA – Supervisory control and data acquisition

From Wikipedia, the free encyclopedia

Supervisory control and data acquisition (**SCADA**) is a control system architecture comprising computers, networked data communications and graphical user interfaces for high-level supervision of machines and processes. It also covers sensors and other devices, such as programmable logic controllers, which interface with process plant or machinery.

https://en.wikipedia.org/wiki/SCADA

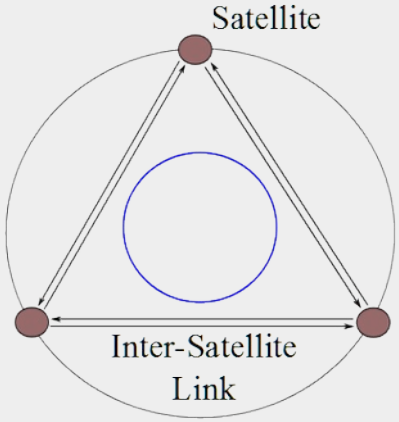
- Industry standard for controlling power plants, production lines, factories, …
- Commercial, very complex tools; vendor-specific protocols;
  require dedicated setup effort
- Requirements are different from experimental physics DCS/DAQ
- Still a lot to learn from (system architecture, nomenclature, nomenclature)

EDDA – Exchange on & Development of DCS/DAQs

...from the project proposal:

- Build a flexible DCS/DAQ software system
- Provide a minimalist interface towards the devices
- Base on industry-standard open source libraries
- Keep required maintenance as low as possible

State of things: Constellation



- We have been struggling with current community solutions for quite some time
  - Used software frameworks devised & written decades ago
  - Partially unstable, unsuited for the task at hand, lack of features
  - Maintenance tedious, technical foundation outdated, code quality degraded over time
- Pondering for a while to replace with a newly-designed framework
  - First works by **S. Lachnit** as part of his master thesis:
    Exploration of structures, libraries to use, implementation of proof-of-concepts
  - Culminated in collection of prototypes & protocols, called **Constellation**
- Hoping that EDDA will meaningful contribute to development

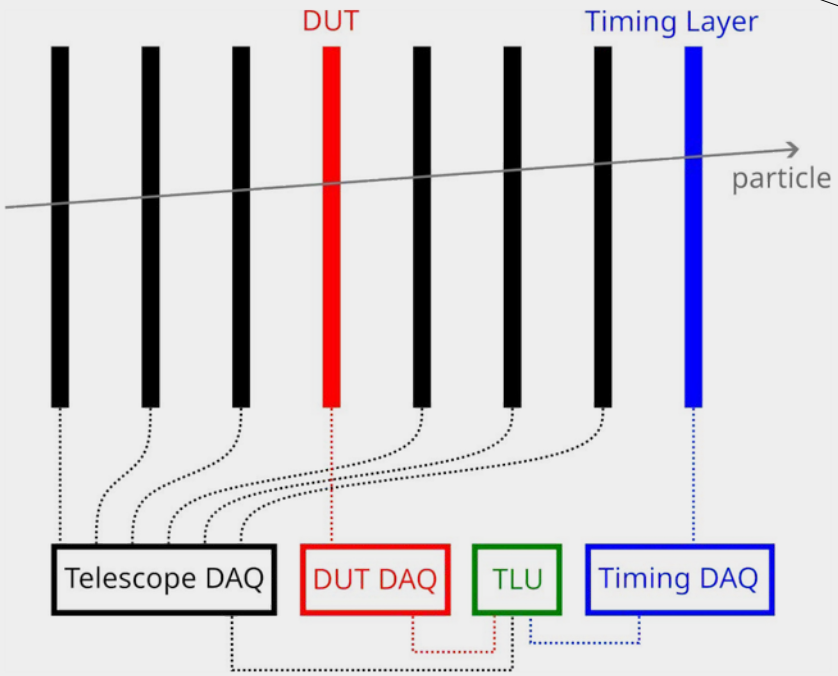
Re: Build a flexible DCS/DAQ software system

- **Useful to control single laboratory setup** (e.g. radioactive source measurement)
  User story: Start measurement in the lab, return to office. Open up monitoring tool, observe data trickling in, temperature being stable. Check logs for anomalies. Go to meeting. Receive phone notification about finished measurement, abort meeting.
- **Possibility to integrate multiple setups** (Detector DAQ, TCT laser control)
  User story: Write common configuration file detailing detector parameters and laser driver/stage configuration. Start measurement.
- **Lab supervision mode** (multiple setups monitored but control not ceded)
  User story: Central monitoring of multiple lab setups, but not necessarily controlled via common Control Center but from individual lab workstations
- **Synchronized operations** (test beam environment, coordinated start/stop, central control)
  User Story: Run multiple systems in parallel with a central control; automated parameter scans, resilience to failures like graceful restart
- **Scalability for experiments** (many detectors, multiple data endpoints & monitors)

Example: Environment in Test Beam Campaigns

Slide by S. Lachnit

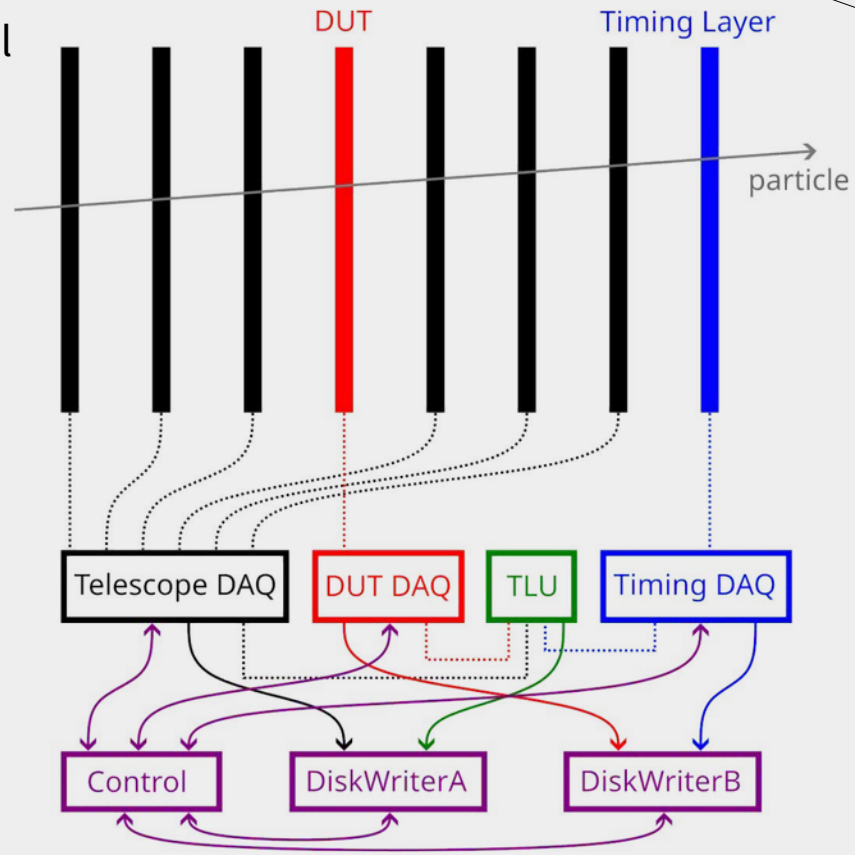
- Testing of new detectors in beam usually requires reference detectors
- Example Pixel Sensor:
  - Device under test (DUT)
  - Telescope for reference tracking
  - Timing Layer for triggering
  - Trigger Logic Unit (TLU) to distribute triggers to Telescope & DUT
- Temperature sensors, ...
- Other detectors (e.g. calorimeters) might require different reference setups



Detector Control & Data Acquisition System

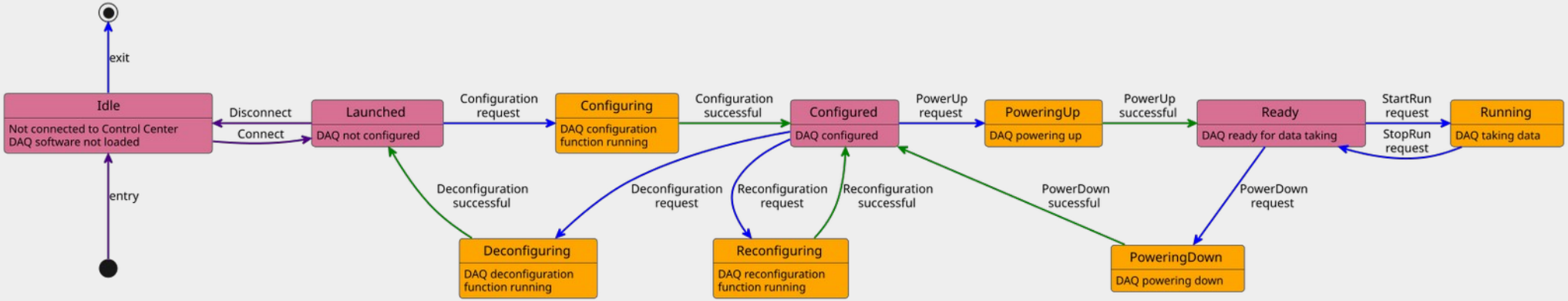
Slide by S. Lachnit

- Need for a central control software to control individual DAQ systems during test beam campaign
- **Tasks**:
  - Start and stop data taking
  - Configure DAQ systems
  - Connect DAQ to storage via network
- **Stable operation** and error handling
  - No crashes in the middle of the night
- **Easy to use** during test beam campaign
- **Straight-forward integration** of new detectors & DAQ systems to the framework



Re: Provide a minimalist interface towards the devices

- Making it easy to integrate devices on short time scales
- Abstraction of communication, heartbeating, log distribution etc in library
- New integrations only required to implement finite state machine transitions:



Conceptual FSM for satellites (S. Lachnit)

Orange: Function to be implemented by module
Blue: User request / supervisor command
Green: Transition after successful user action
Purple: Automatic transition from framework

Re: Base on industry-standard open source libraries

- Maximize data throughput, robustness and configurability

- Today, many network communication libraries available, no need to roll our own (anymore)
  - Messaging, message queuing, routing, …
  - Data packaging, binary wire representation, cross-platform portability, …
- Monitoring tools from networking context
- Modern infrastructure (build system, version control, CI/CD)

Example: Grafana Dashboard

Concept: Robustness and Resilience

- Not all failures should lead to an abortion of the current session
- What should be done if a satellite crashes in the middle of a run?
  The concept of **importance**
  - **essential:** run can not be started without started without the Satellite, run is immediately stopped on connection loss / interrupt
  - **desired:** run is marked as degraded if Satellite not connected or connection loss / interrupt during run, but does not cause a run to be stopped or prevents a run from being started
  - **regular:** only marks run as degraded when added or removed during run
  - **optional:** does not affect run in any way

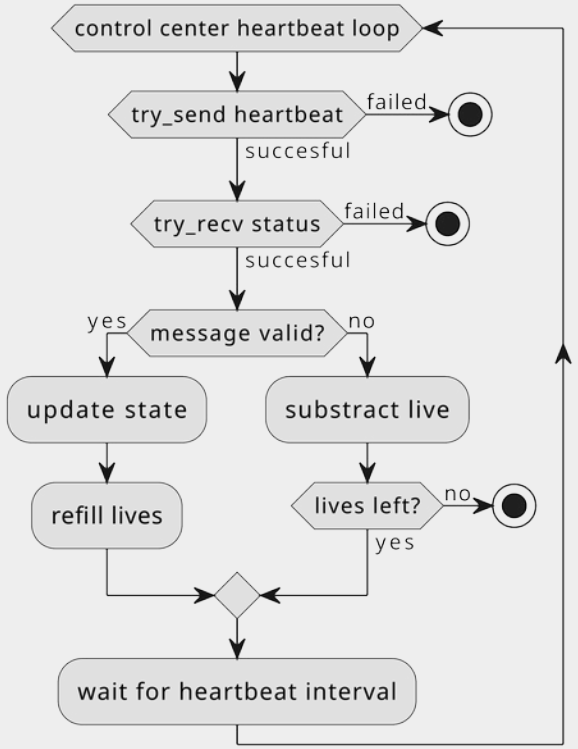
Concept: Robustness and Resilience

1) Each satellite can has user-configured *importance*
2) Action depends on level of *importance*
3) If the satellite importance is **essential**, the run will be aborted
   If the satellite importance is **desired, regular or optional**, the run is not aborted
4) Current run/session marked as **degraded** if importance is higher than *optional*
   (akin to a *degraded RAID system*)
5) Alarm Handling – Notification to the user (UI, Email, Mattermost, etc.)

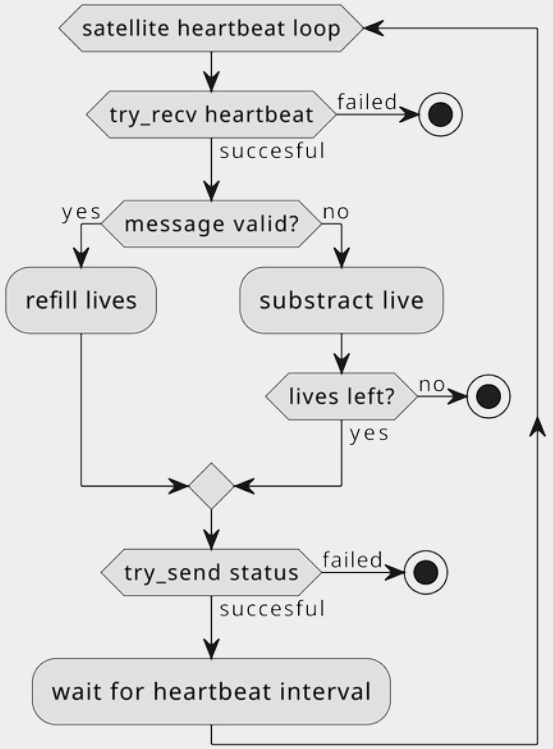
Concept: Heartbeating

- Continuous low-frequency message pattern
- E.g. implemented as ping-pong (REQ-REP) pattern
- Ensure components of constellation are still connected
- Built-in lenience for congestion situations (multiple lives)

Control Center:



Satellite:



Concept: Logging / Monitoring

- Flexible & transient subscription to log levels & devices
  - During run, new logger can be started, which subscribes to a given log type
  - Log type subscription can be changed on the fly
- Based on ZeroMQ RFC 29/PUBSUB, allows to publish messages with specific topic
  - Subscribers can subscribe to multiple topic via pattern matching
  - Publish logs e.g. to <LOG_LEVEL>/<SAT_NAME> and stats to <SAT_NAME>/<STAT>
  - Loggers can subscribe to global log level as well as log levels for specific Satellites
- Network protocol ensures that there is only traffic on publisher when there is a subscriber to the topic
  - Device code can contain 1000s of DEBUG message statements, they never leave the machine unless someone requested them.

Re: Keep required maintenance as low as possible

- Document everything!
  - User manual describing setup & operations
  - Developer manual detailing code base structure & environment
  - Explicitly documented, versioned network protocols for communication

- Essentially follow industry-standards on coding conventions:
  - Work with strict requirements (code formatting, linting, code documentation)
  - Merge code after code reviews from other developers only
  - Extensively use continuous integration & deployment (GitLab CI/CD)

Concept: CHIRP – zeroconf network discovery

Constellation Host Identification and Reconnaissance Protocol

- Status: draft
- Editor: The Constellation authors

The Constellation Host Identification and Reconnaissance Protocol (CHIRP) defines how different hosts announce their services and connect to each other on the network.

Preamble

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Goals

The CHIRP protocol provides a way of discovering and announcing services on a network segment and provide the means for hosts to connect over peer-to-peer connections. The goals are:

- To work with no centralized services or mediation except those available by default on a network.
- To allow service discovery both for late-joining clients and for lingering clients when a service provider host appears late.
- To facilitate the exchange of connectivity information for different services and the ability to select distinguish between them.

Implementation

Identification and Life-cycle

A CHIRP host represents a source or a target for messaging. Hosts usually map to applications. A CHIRP host is identified by a 16-octet universally unique identifier (UUID). CHIRP does not define how a host is created or destroyed but does assume that hosts have a certain durability.

Host Discovery and Service Announcement

CHIRP uses UDP IPv4 beacon broadcasts to discover hosts. Each CHIRP host SHALL listen to the CHIRP discovery service which is UDP port 7123. Each CHIRP host SHALL broadcast, upon creation, on UDP port 7123 a beacon that identifies itself to any listening hosts on the network. A separate CHIRP beacon SHALL be broadcast for every service the host advertizes.

The CHIRP beacon consists of one 27-octet UDP message with this format:

```
+---+---+---+---+---+------+  +------+------+---------+------+
| C | H | I | R | P | %x01 |  | UUID | type | service | port |
+---+---+---+---+---+------+  +------+------+---------+------+

          Header                          Body
```

The header SHALL consist of the letters ‘C’, ‘H’, ‘I’, ‘R’ and ‘P’, followed by the beacon version number, which SHALL be %x01.

The body SHALL consist of the sender’s 16-octet UUID, followed by a one-byte beacon type identifier, a two-byte service descriptor, and a two-byte port number in network byte order. If the port is non-zero this signals that the peer will accept ZeroMQ TCP connections on that port number. If the port is zero, this signals that the peer is disconnecting from the network.

The type SHALL be either %x01 (dubbed ‘REQUEST’) or %x02 (dubbed ‘OFFER’).

A valid beacon SHALL use a recognized header and a body of the correct size. A host that receives an invalid beacon SHALL discard it silently. A host MAY log the sender IP address for the purposes of debugging. A host SHALL discard beacons that it receives from itself.

When a CHIRP host receives a beacon of type ‘OFFER’ from a host that it does not already know about, with a non-zero port number, it MAY connect to this peer if it SHOULD participate in the offered service.

When a CHIRP host receives a beacon with type ‘REQUEST’ from any host, with a zero or non-zero port number, and it offers the requested service, it MUST respond with a CHIRP beacon of type ‘OFFER’ for the requested service.

When a CHIRP host receives a beacon of type ‘OFFER’ from a known host, with a zero port number, it SHALL disconnect from this peer.

Protocol Grammar

```
chirp   = [request] *offer

; Request offers from other hosts
request = header version uuid %x01 service port

; Make an offer of a service to other hosts
```

Summary of Status Quo

- Have started collecting concepts & protocol ideas for new DCS/DAQ/SCADA framework for small experimental physics lab setups & experiments
  - First concepts for communication & protocols
  - First ideas for FSM states & transitions
  - Some concepts for log distribution, monitoring, UI

- Still early phase, actively looking for contributors
  - Development of the software
  - Contributions in the form of sketches, ideas, requirements, problems to be solved
  - Later on: testing

Ideas Worth Exploring

- Do we need a constantly running Control Center or can we make an entirely distributed system work? (distributed FSM)
- What are options for writing UIs / HMIs (Human Machine Interfaces)
- What situations does the FSM (finite state machine) need to be capable of handling? (Disconnects, OOM, disk full, config errors, …)
  What Alarm Handling modes do we need (notification, auto-reconfigure, …)
- Your ideas, your requirements

- Fill the questionnaire: https://t1p.de/kv8u1


DESY.