KM3NeT Acquisition Control

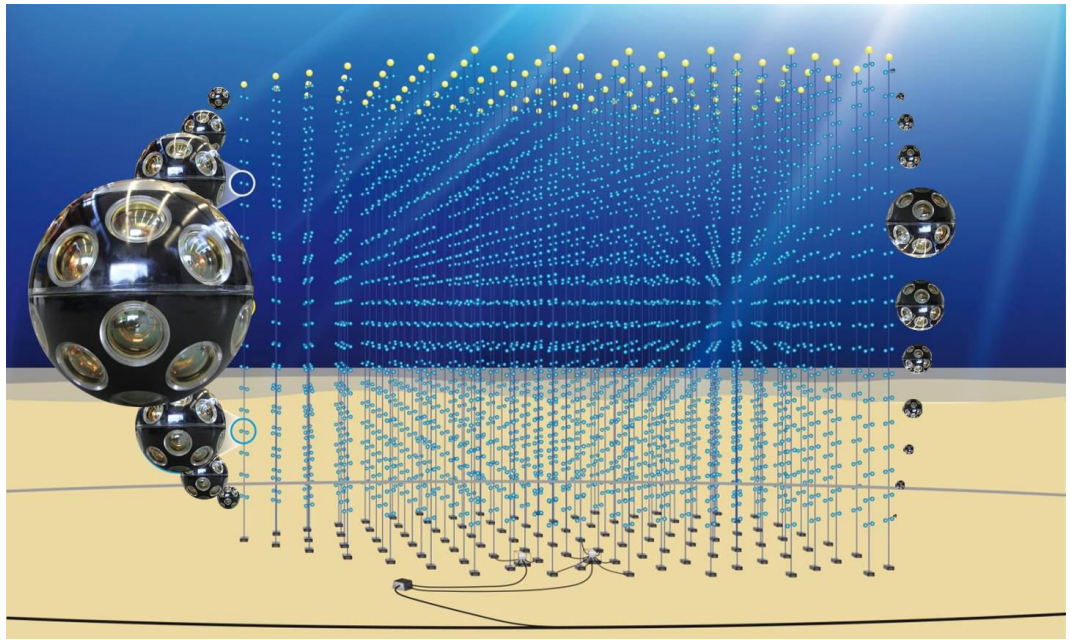
Cristiano Bozza

(University of Salerno and INFN Gruppo Collegato di Salerno)

for the **KM3NeT Collaboration**

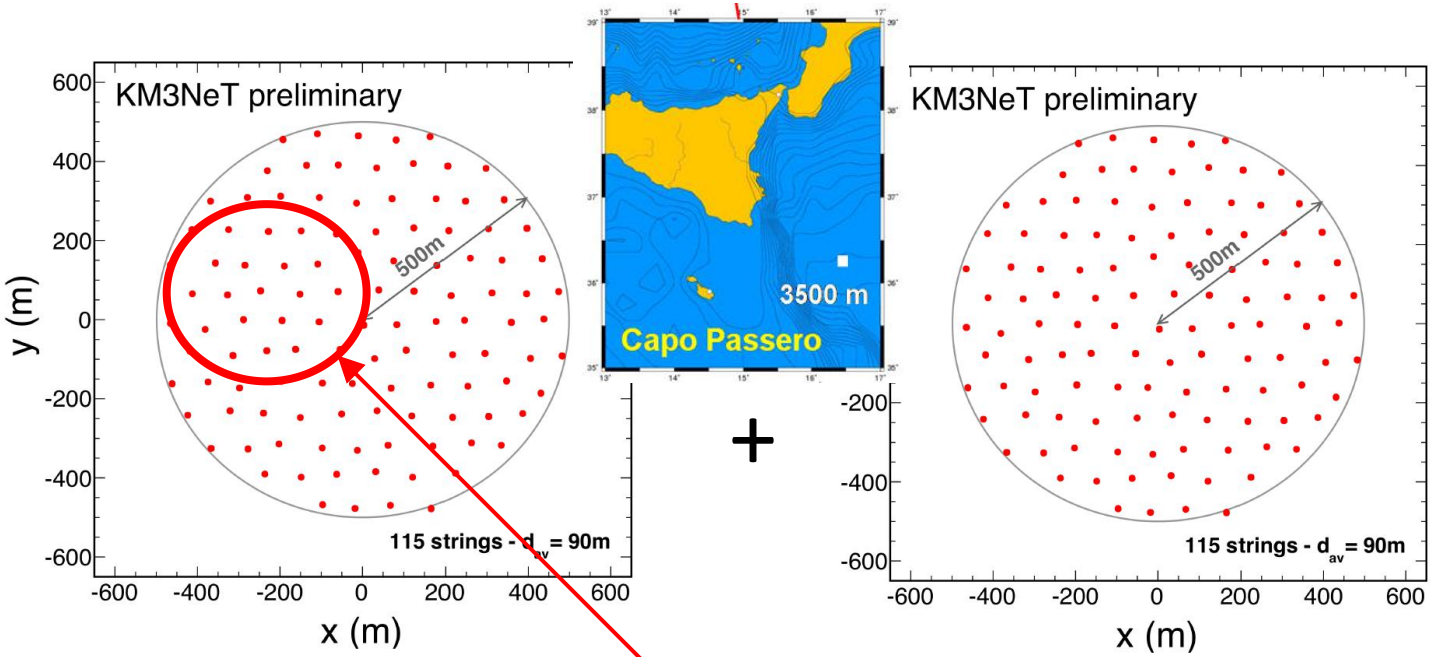
Dubna, JINR, 2018

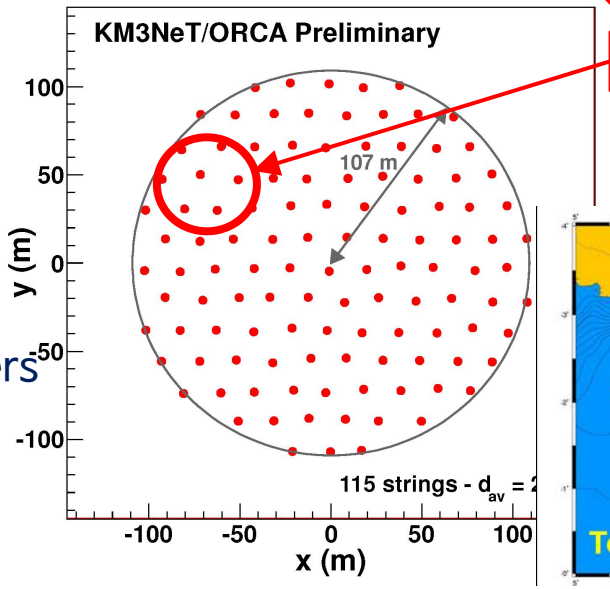
KM3NeT Detectors

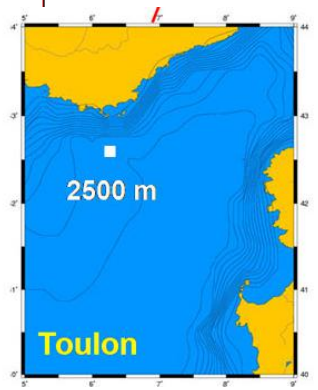


1 building block = 115 DUs
1 DU (detection unit) = 18 DOMs + 1 Base module
1 DOM (Digital Optical Module) = 31 PMT photomultipliers

2185 CLBs (Central Logic Boards) / block
64170 PMTs / block





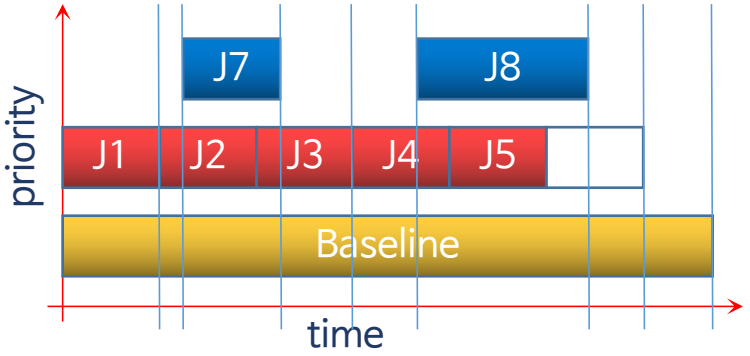


Phase1 (fully funded – deploy 2018/19)

Phase 2 (partially funded – deploy 2019/21)

2 shore stations
+
Several test stations
=
Multisite enterprise
≈10 "detectors" active

Data and control flows



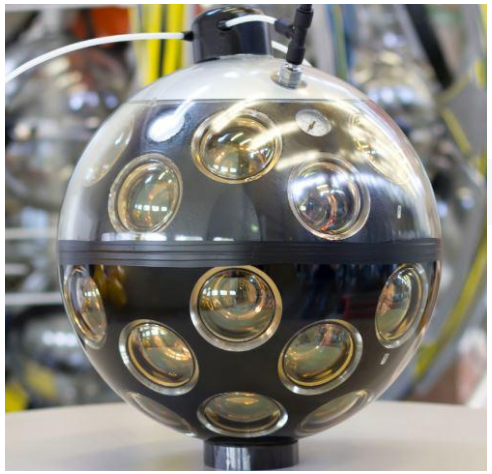
Master Control Program:
Schedule and current data acquisition mode

Run = stable detector configuration and set of parameters for PMTs, instruments and triggers

Jobs → acquisition modes for normal operation + schedule calibrations and special tasks (e.g. transients, tests)

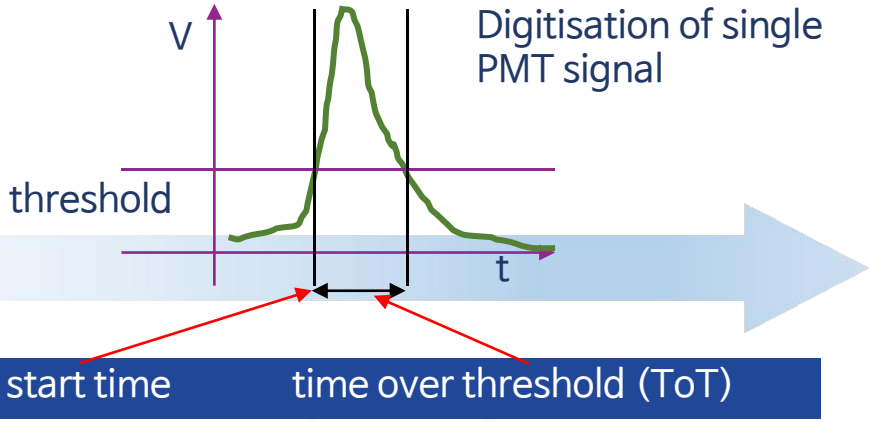
Flexibility in detector operation is naturally supported

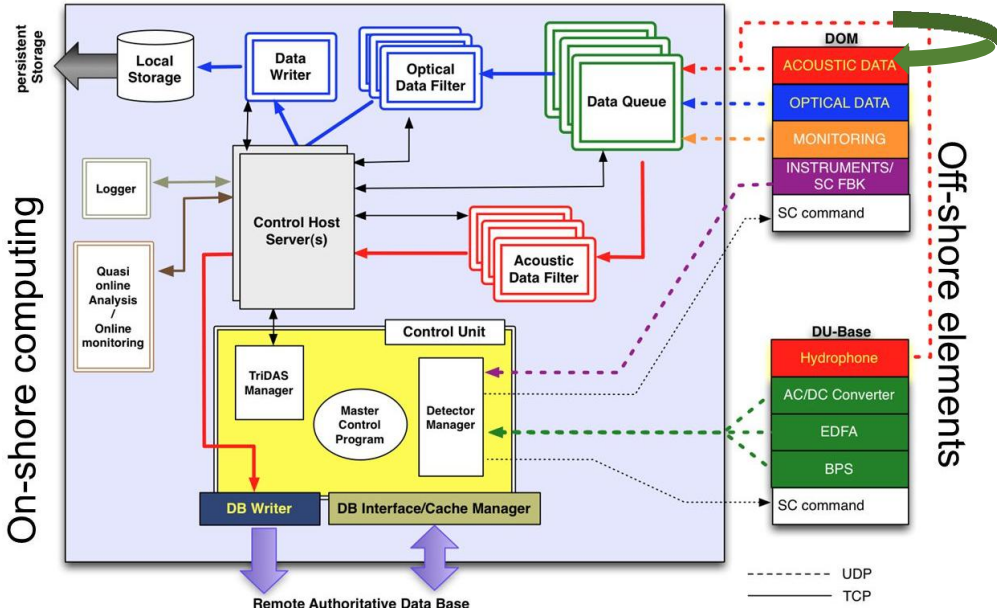
"All data to shore" concept



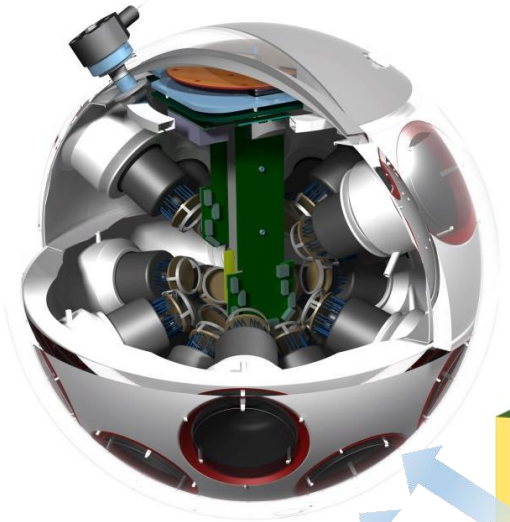
Trigger and DAQ running entirely on-shore

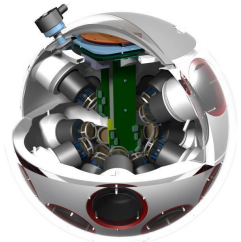
Flexible approach for computing power and algorithms

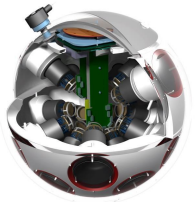




Detector definition and runsetups

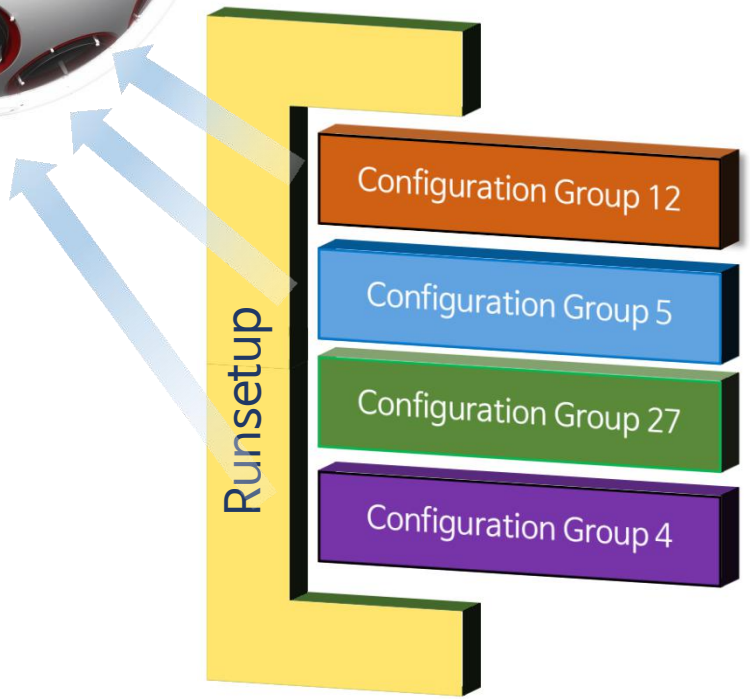






Detector structure defined in database via integration information

Only components relevant to acquisition control are read into the Control Unit cache



Common input parameters (e.g. flags, min HV)

Custom input parameters (e.g. HV by PMT)

Trigger parameters

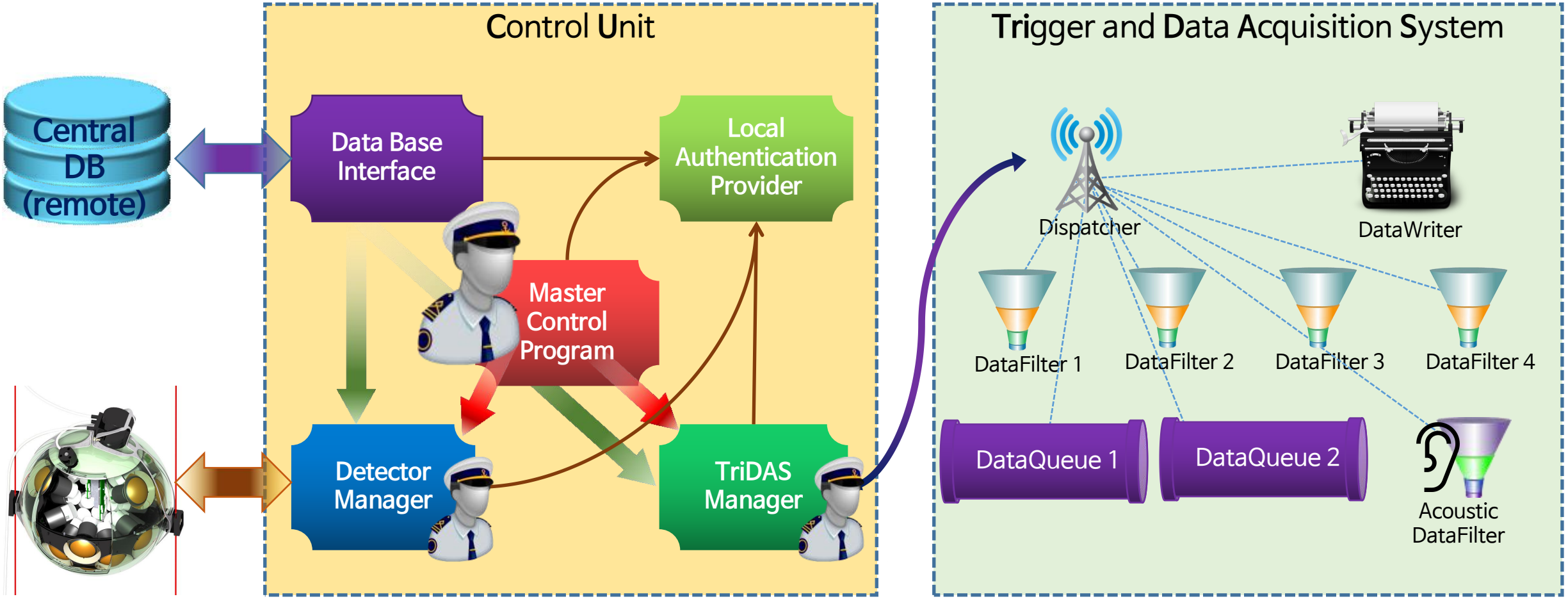
Output parameters (e.g. instrument readout)

Runsetups combine Configuration Groups as "bricks" of settings both for detector and for trigger/acquisition

All information is managed centrally in the DB

Control Unit and TriDAS

Only control-related connections shown

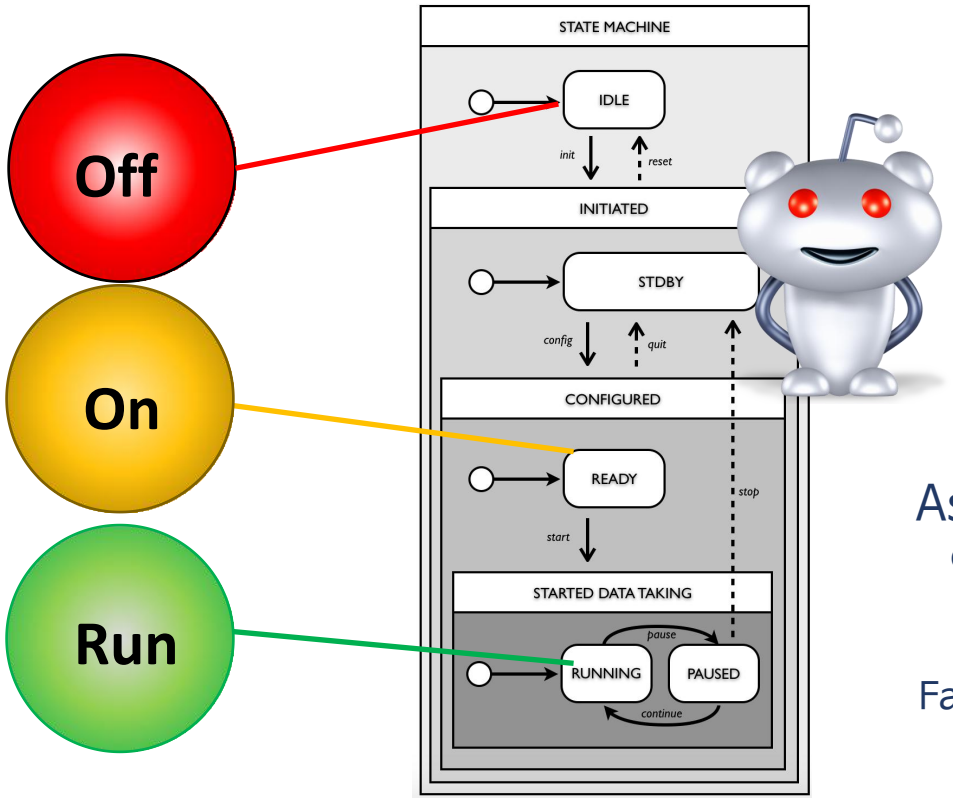


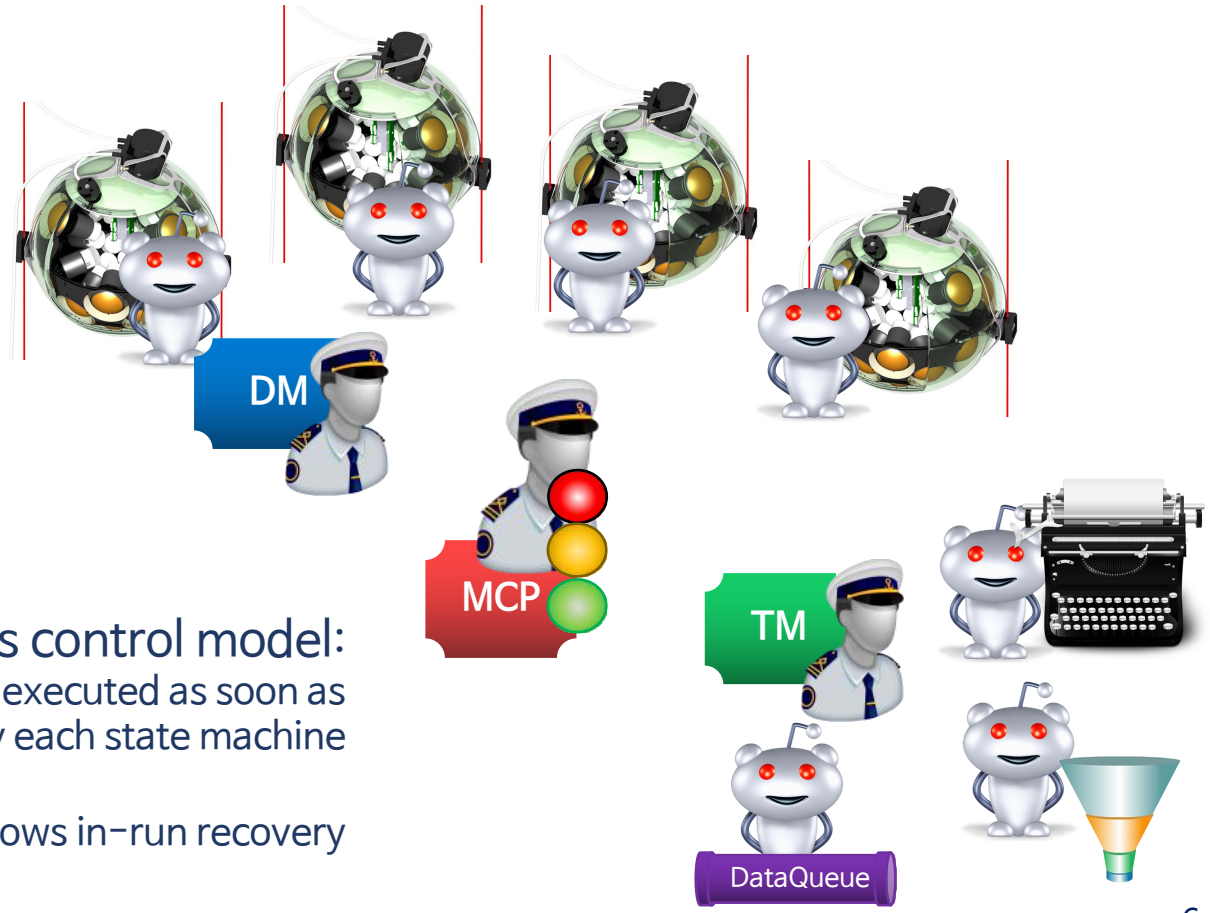
Control logic

- **Off**: all PMTs turned off, no HV
- **On**: all PMTs turned on (HV on), no optical data generated (TDC off)
- **Run**: all PMTs turned on, TDC on to generate optical data

The MCP sets the global "Target"

Finite State Machines at many levels mirror the behavior of real objects



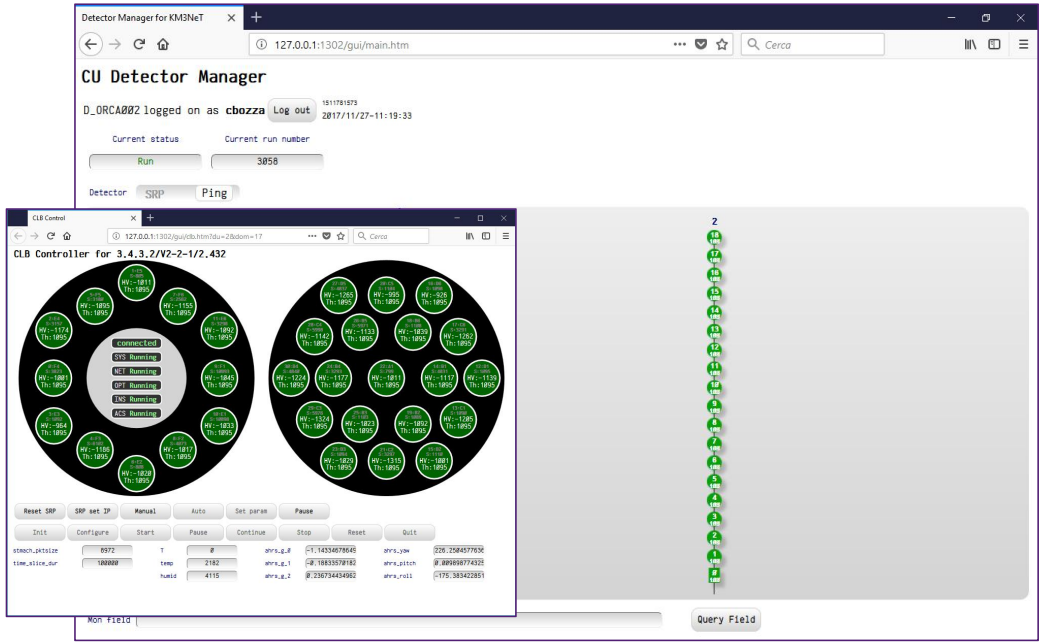


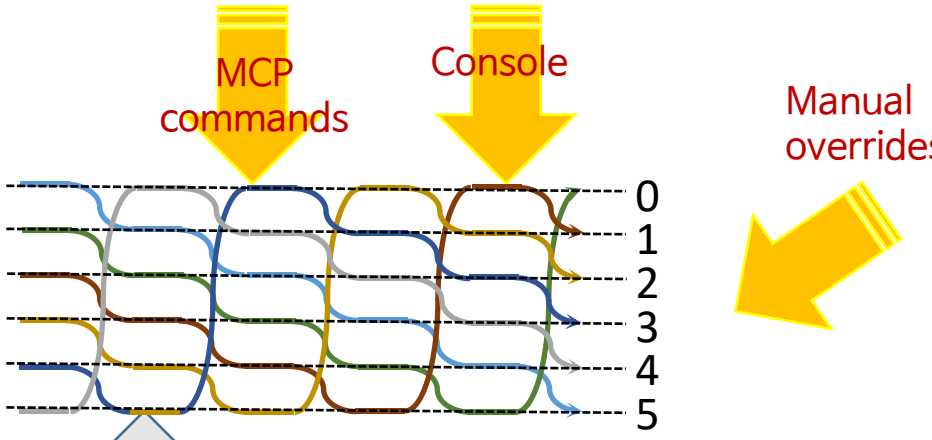
Asynchronous control model:
each "order" is executed as soon as possible by each state machine

Fast, scalable, allows in-run recovery

Detector Manager

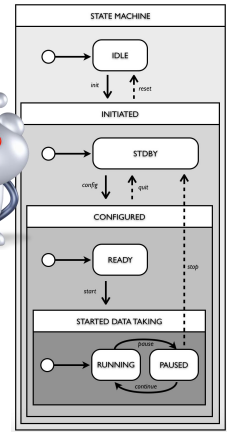
- Drives each CLB (**C**entral **L**ogic **B**oard) to apply configuration settings
- Drives the state machine of each subsystem of each CLB to comply with the target dictated by the MCP
- Monitors each DOM (PMT HV/threshold/rate, temperature, humidity, acceleration, compass…)
- Allows manual override when needed





Threads (coloured paths), created in a controlled number (depending on the number of DOMs), controlling CLBs

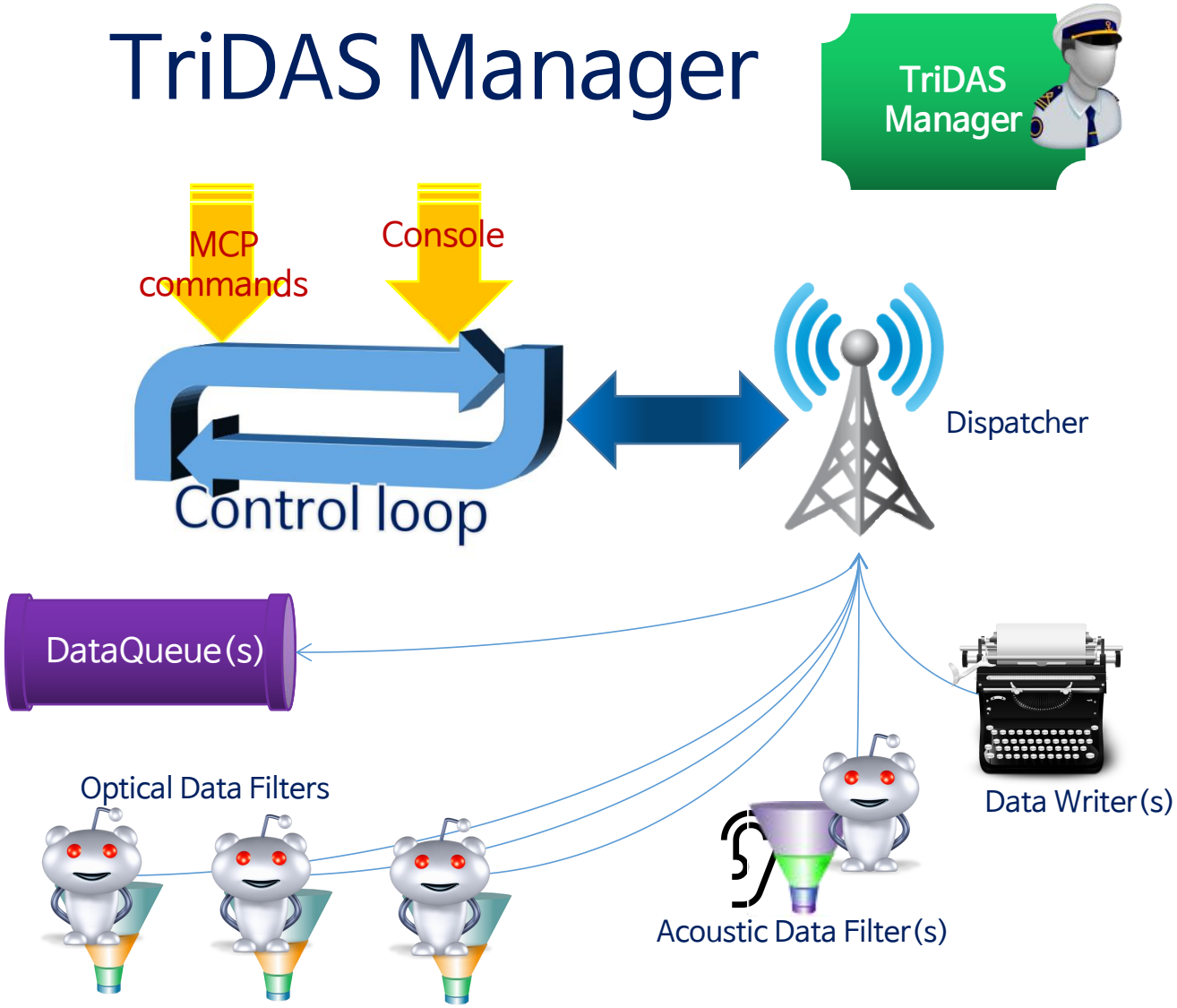
CLB



DM Presents monitoring data to:
- Virtual Directory, e.g. http://myserv.er/mon/outparams/humid/3/6
- Datalog files (binary files for upload to DB)
- Text log files

Estimation: **12 CPU cores** should be able to monitor **115 DUs**

TriDAS Manager

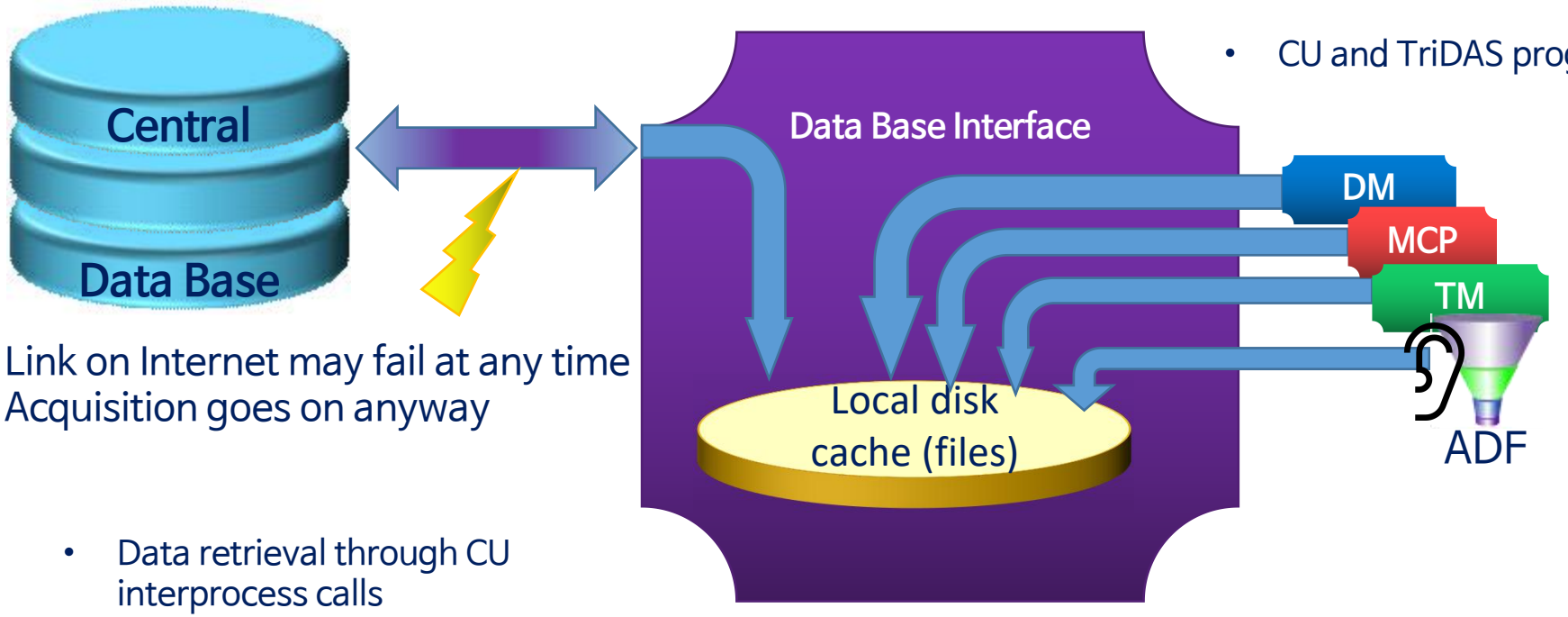


- Drives each **TriDAS process** (DataQueue, optical DataFilter, acoustic DataFilter, DataWriter) to apply configuration settings
- Drives the state machine of each TriDAS process to comply with the target dictated by the MCP
- Monitors each process, initiating restoration actions if needed
- Reads all state change events from Dispatcher as a serial stream, writes reactions to the Dispatcher as a stream
- Presents monitoring data to:
  - Virtual Directory, e.g.
    http://myserv.er/mon/dq/states/DQ/2
  - Datalog files (binary files for upload to DB)
  - Text log files

Data Base Interface

- Allows easy retrieval of data from the central Data Base
- All data relevant to detector operation are synchronized to a local cache that stores XML files

- Data to be written to the DB are staged in the local cache
  Writes are retried in case of link failure
  MCP jobs and runs
  DM detector monitoring and detector definition changes
  TM TriDAS monitoring and detector definition changes
  ADF Time of Arrival of acoustic signals for positioning
  LAP dynamic provisioning and failover actions (see next slides)
- CU and TriDAS programs write directly to the stage directory

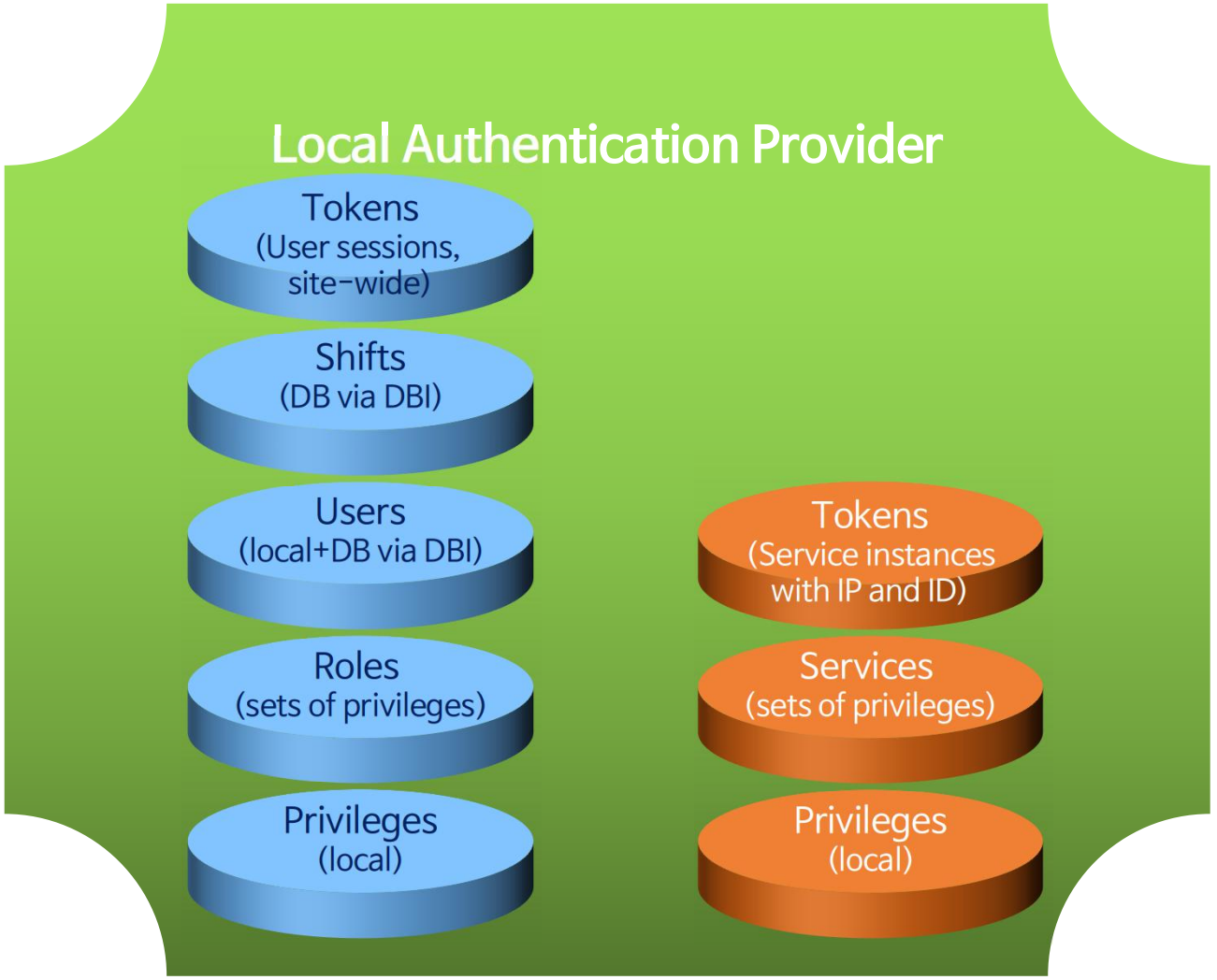


Link on Internet may fail at any time
Acquisition goes on anyway

- Data retrieval through CU interprocess calls
- DBI also notifies MCP of availability of new data

Authentication, Identification, Roles, Privileges

- User privileges
- DB users synchronized with local cache
- Roles = privilege set
- Shifts → Temporary role assignments for users
- Tokens = site-wide user session



- Service privileges
- Services = functions (MCP, DM, TM, etc.)
- Token = incarnation of service with host IP, port, executable, identity key
- A service may be moved to another host or use a different (version of) program

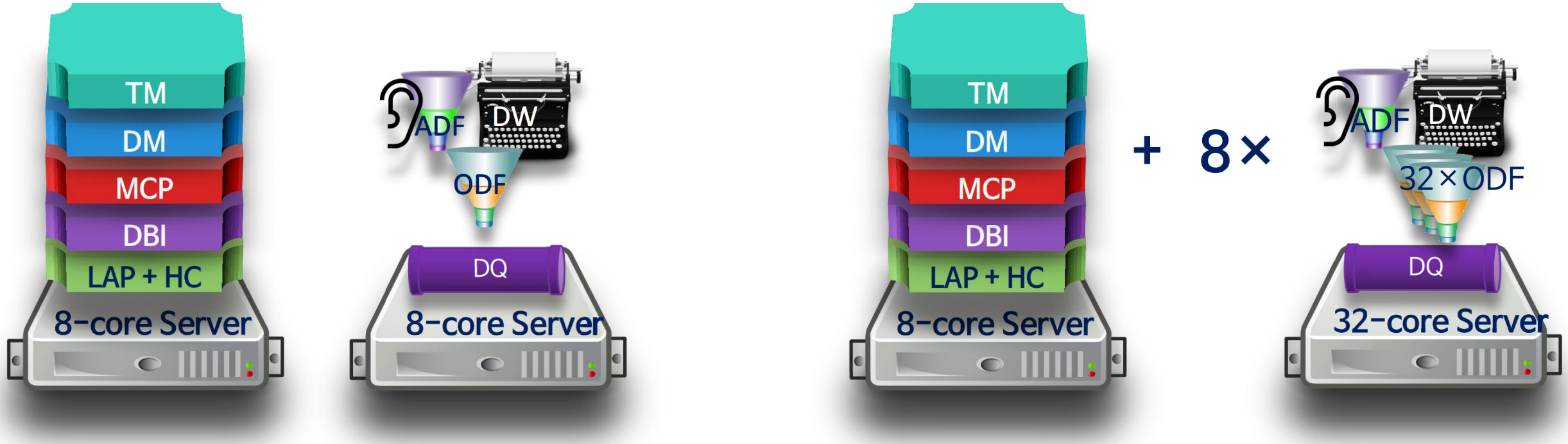
Resource Allocation

The same control and DAQ software is used in the shore stations of physics detectors and in test/qualification stations

A single CU server with 32 cores should have enough power to control two building blocks (total 230 DUs)
Data logging is the CPU-consuming part

Minimal installation, test/qualification station

About 100 cores are expected to be needed for triggering and data preprocessing of one building block
Machinery for two blocks



Dynamic Resource Provisioning and Failover – 1

Full DAQ infrastructure for KM3NeT → many machines, hundreds of cores, tens of network interface cards

Machine replacement/insertion → system reconfiguration

Impact of hardware failures

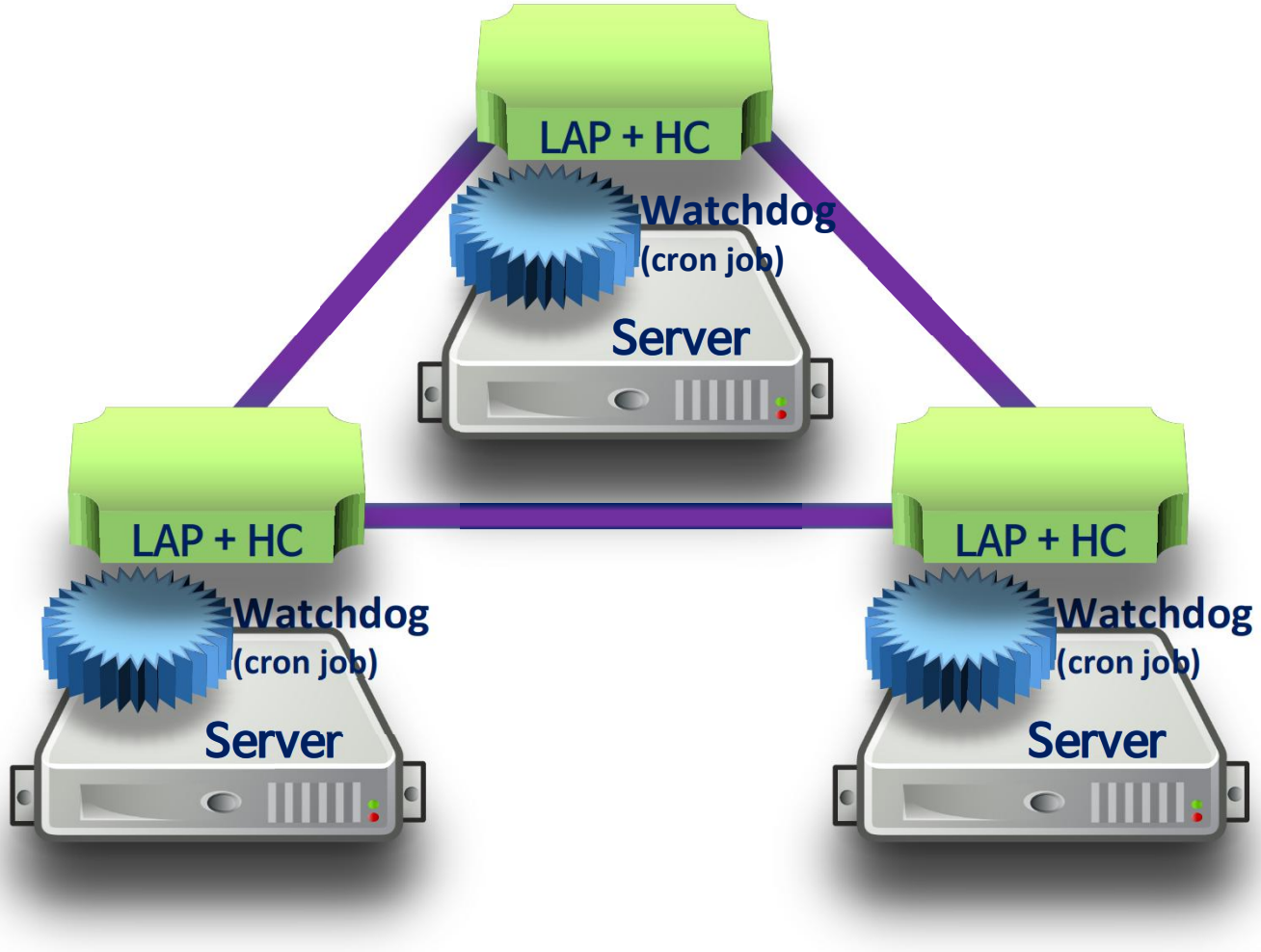
- Failure of processing machines → partial data loss
- Failure of control machines → operation stop (total data loss)
- Administrators might not be available to help during a transient phenomenon
- The system has the resources to work in sub-optimal configurations

One LAP per machine, mutual synchronisation

A **H**ealth **C**hecker (HC) function is added to LAPs, polled every few seconds

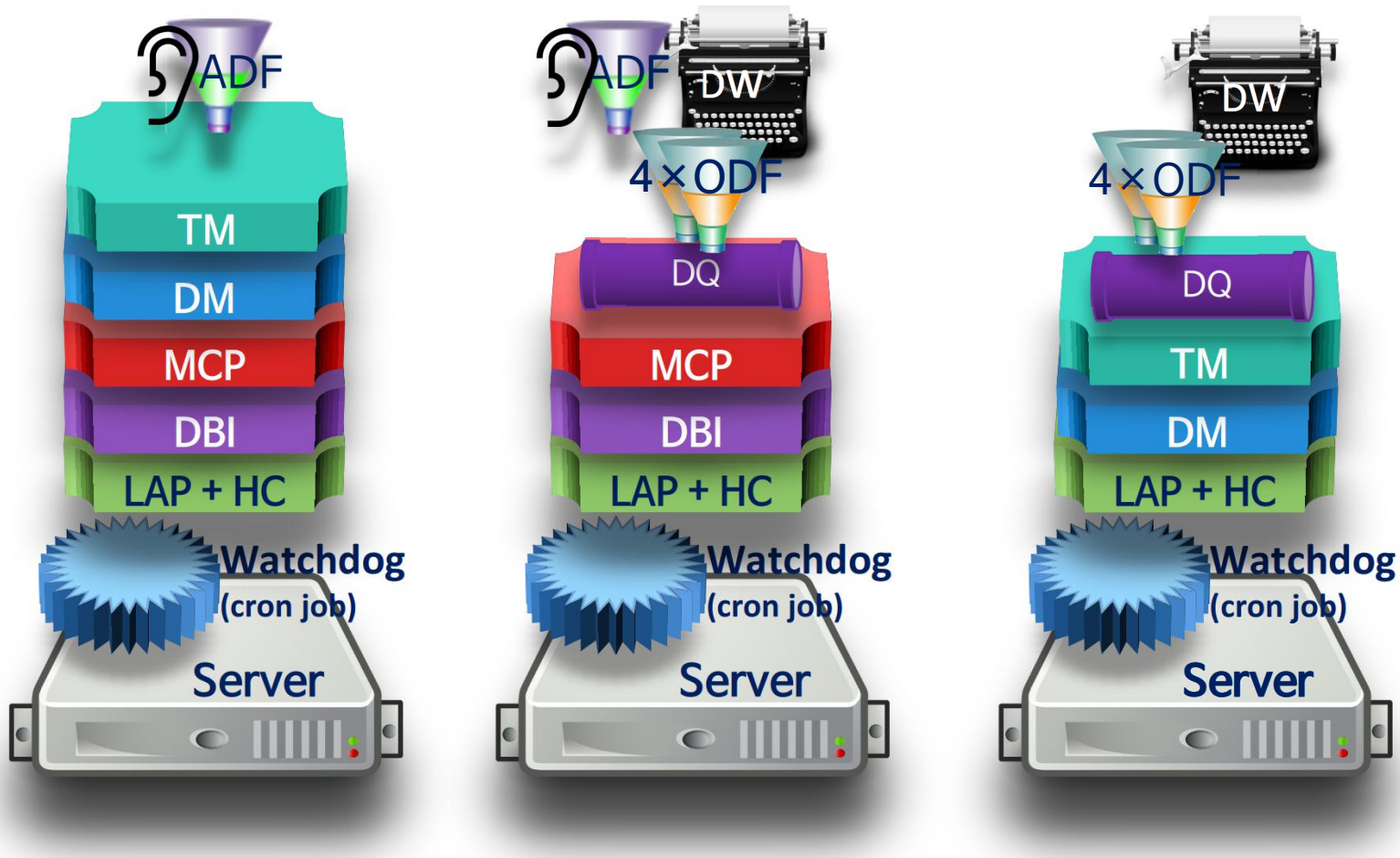
Failure to answer or returning an exception would set the machine offline:

- peer servers know it failed
- failed machine, if still running, knows it's not operational



Dynamic Resource Provisioning and Failover – 2

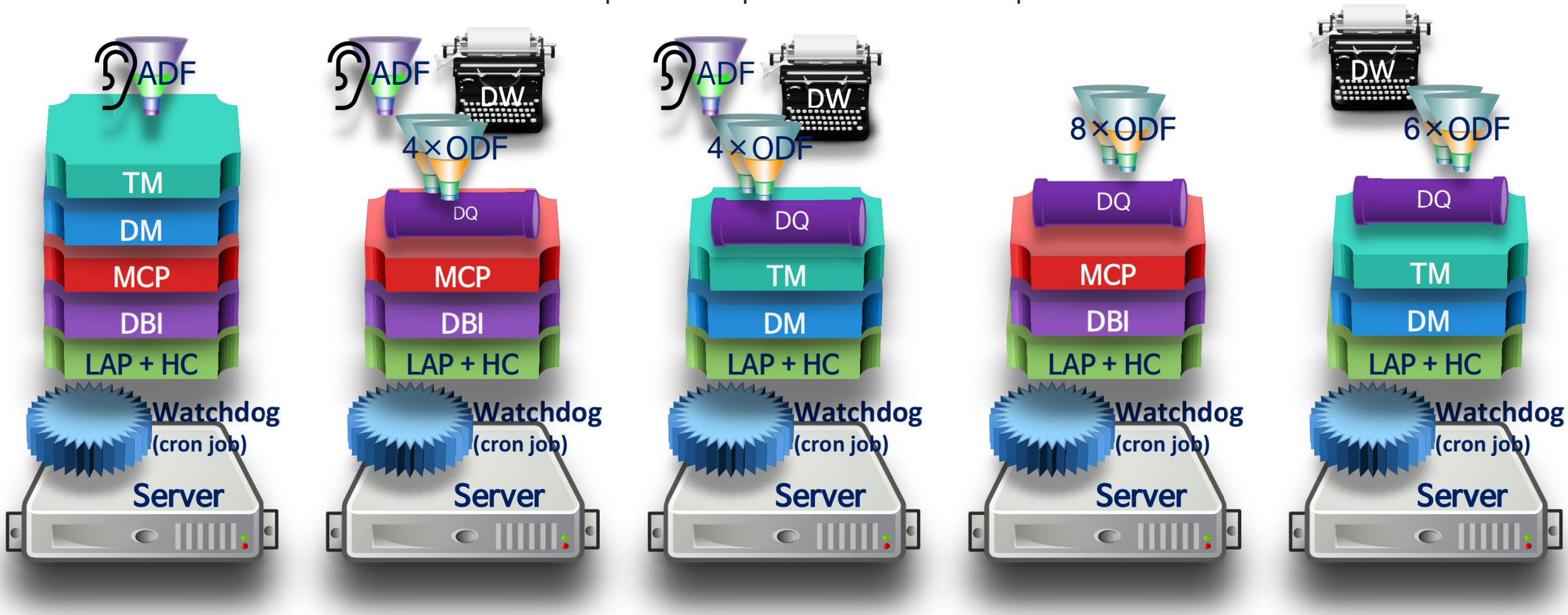
Example of single-fault tolerant setup



- DM/TM are notified when the list of operational machines changes
- All servers have one LAP (+ HC)
- Only one active MCP, DBI, DM, TM: the role is taken by the machine with the lowest IP **(no central authority!)**
- Automatic synchronisation → register a new machine against one LAP (sync will do the rest)
- DRP-F simplifies administration

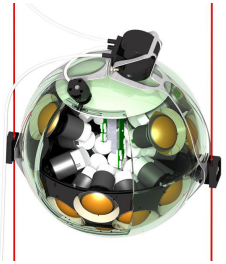
Dynamic Resource Provisioning and Failover – 3

Example of multiple-fault tolerant setup

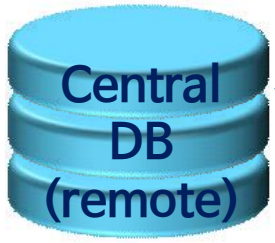


(this picture only shows the logical scheme: server cores and number of ODFs not scaled to 115 DUs)

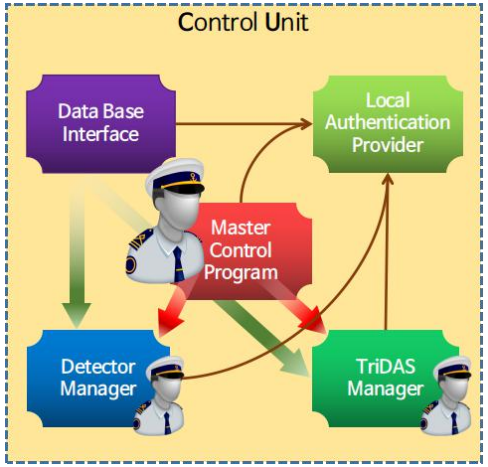
Networking



SRP (UDP)
In-house built UDP protocol with payload optimisation and possibility of retransmission of lost packets

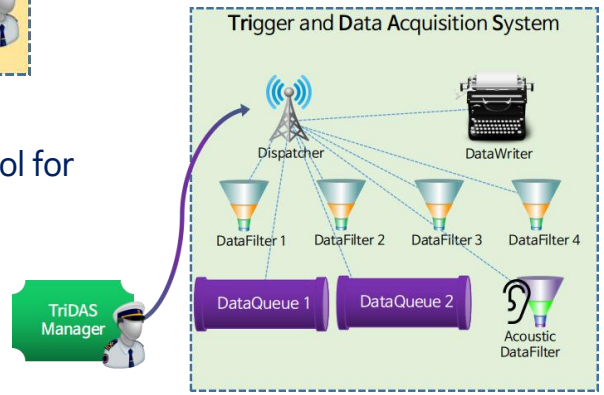


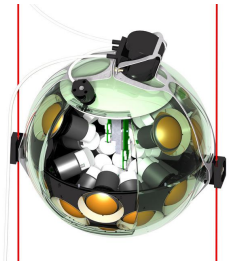
SQL*Net (TCP)
Oracle standard protocol for server-client communication over LAN and WAN

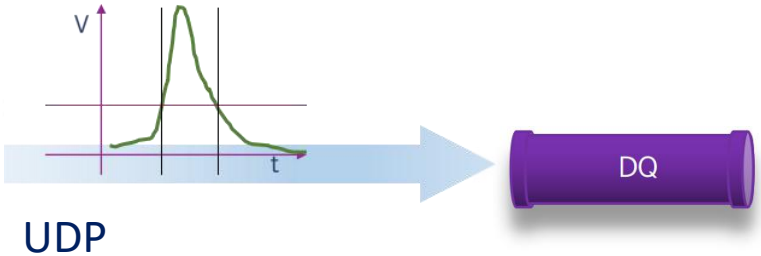


SAWI (HTTP) for RPC
Relies on in-house built webserver as a library, not host process
+
NFS for datalogs

Control Host (TCP)
Connection-oriented protocol for message passing
+
NFS for data writing







UDP
Data packets from CLBs (PMTs, Acoustic channel)

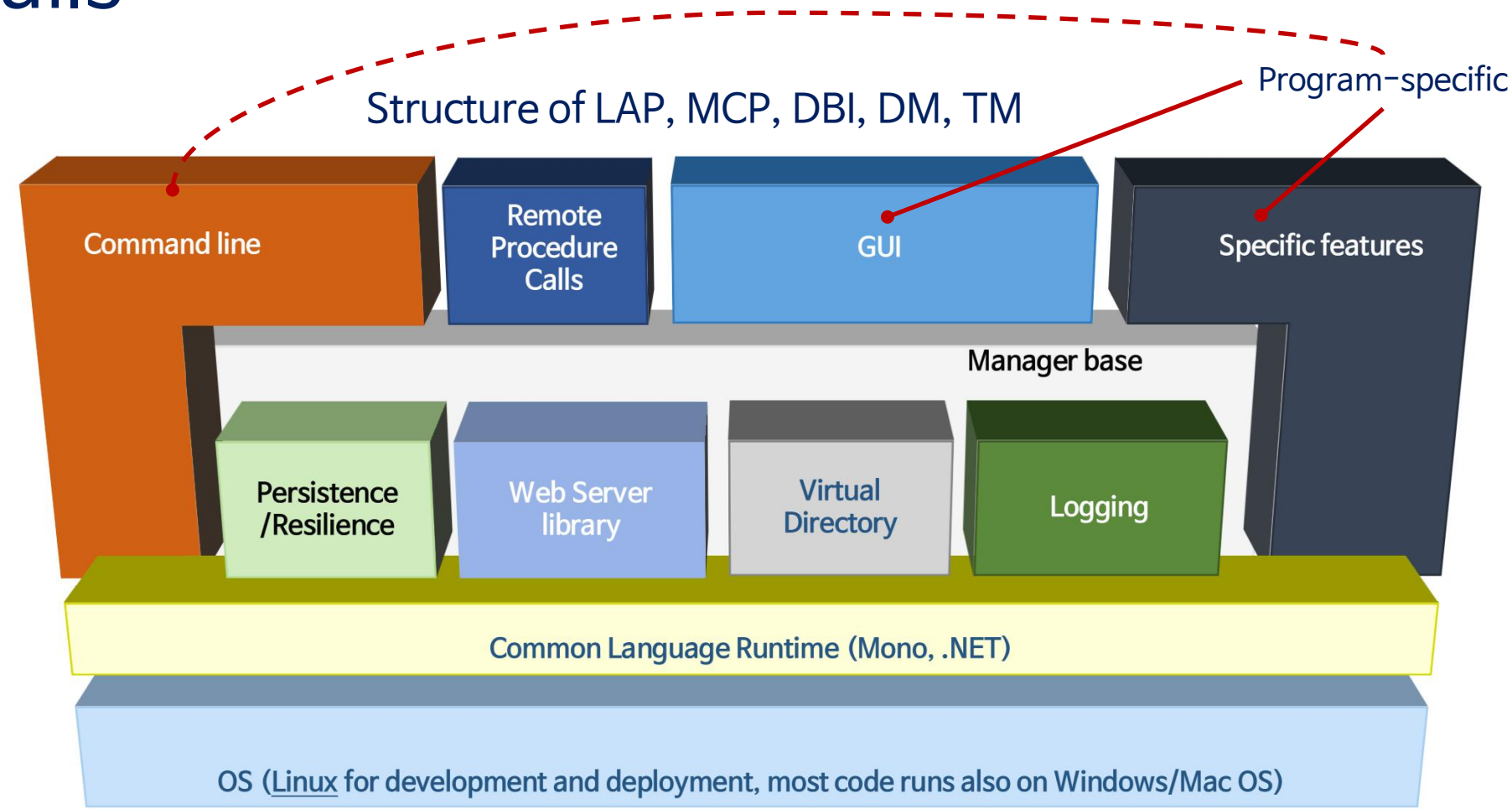
Coding details

All code written in C#

Requires Mono 5.x and depends on no other external library

Runs on CentOS 7
(but at least also on:
Ubuntu 16.04
Ubuntu 18.04
Debian 8
Debian 9
LMDE 2
Mint 18
Mint 19
SLC 6)

Also successfully tested in Docker containers



Code base is largely reusable for upgrades and possibly other projects

Conclusions

- The KM3NeT Acquisition Control software is running detectors as well as test/qualification benches
- Maximises detector live time, but also aims at being user-friendly and handle most situations automatically
- "Maximally disconnected" architecture: each service can run for finite amounts of time without interactions with the others
- Asynchronous control mode allows economy of hardware resources, relatively simple coding and ability to recover single devices/processes without stopping runs
- Recent developments: simplify long-term management and recovery from abrupt hardware failures
- The project leverages modern technologies such as C#, Mono and Web development for the GUI and Remote Procedure Calls
- Modular structure, code base flexible and ready for updates/upgrades and other uses

Thank you for your attention!