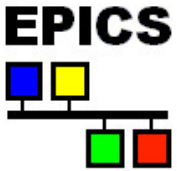
Possibilities for communication between EPICS and TRBnet/DIM

Florian Feldbauer

Helmholtz-Institut Mainz
Johannes Gutenberg-Universität Mainz

DAQT/FEE Workshop
April 10, 2015

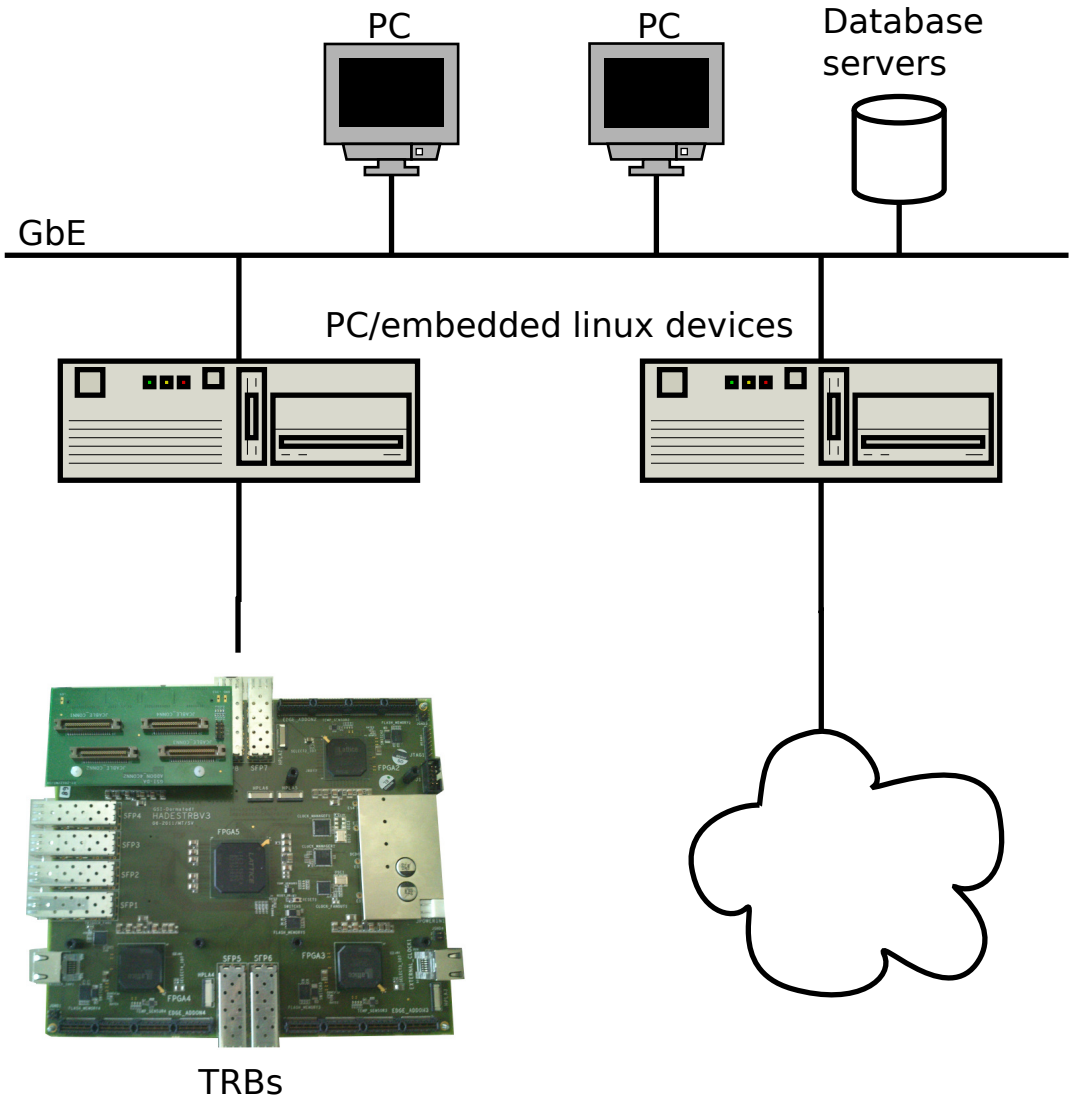
What is EPICS?



The Experimental Physics and Industrial Control System (EPICS) is:

- A Collaboration
  - A world wide collaboration that shares designs, software tools, and expertise for implementing large-scale control systems
- A Control System Architecture
  - A client/server model with an efficient communication protocol (Channel Access) for passing data
  - A distributed real-time database of machine values
- A Software Toolkit
  - A collection of software tools collaboratively developed which can be integrated to provide a comprehensive and scalable control system

Canonical Form of an EPICS Control System



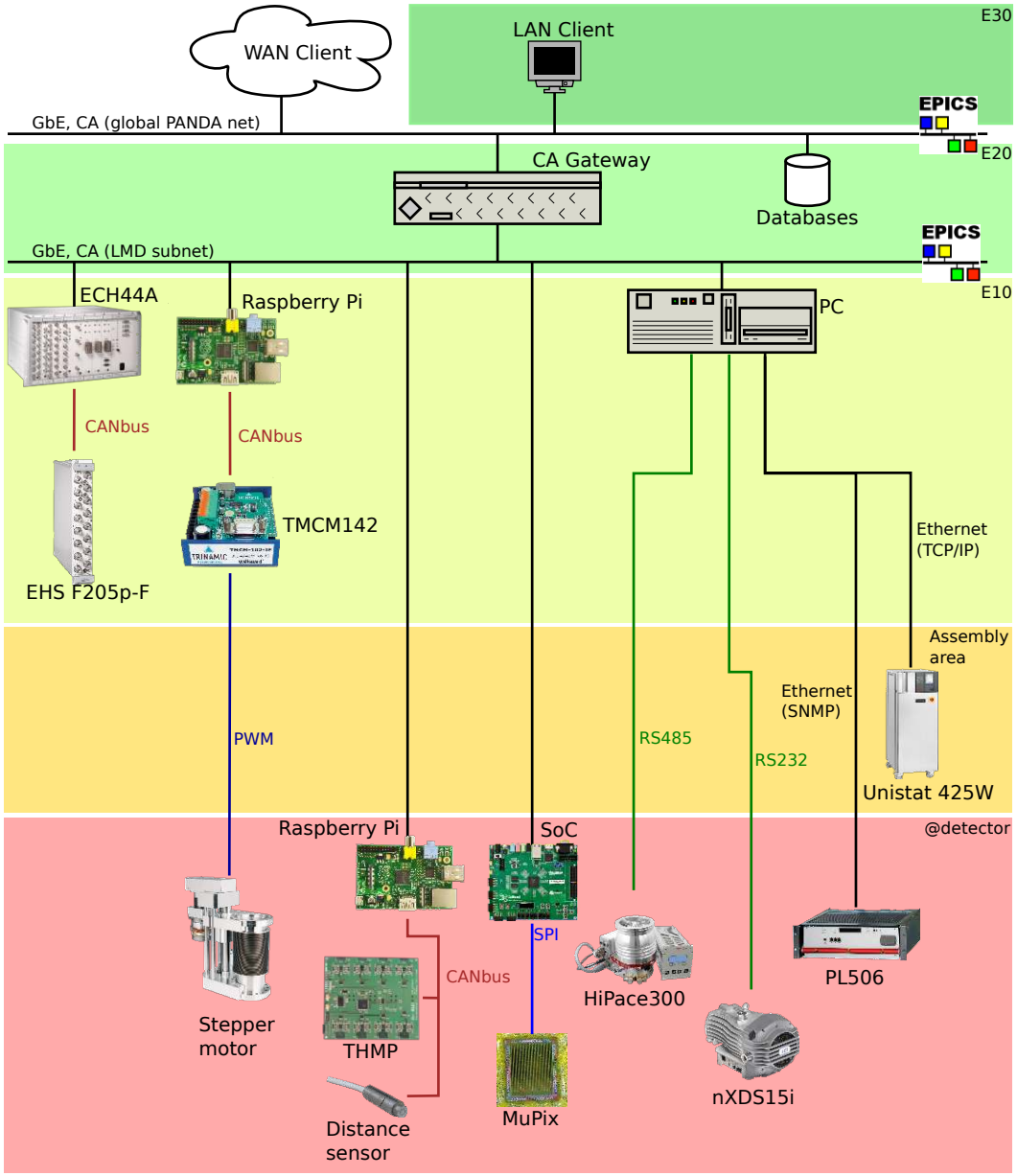
PCs running CA Clients
(Archiver, GUIs, AlarmHandling)

Channel Access

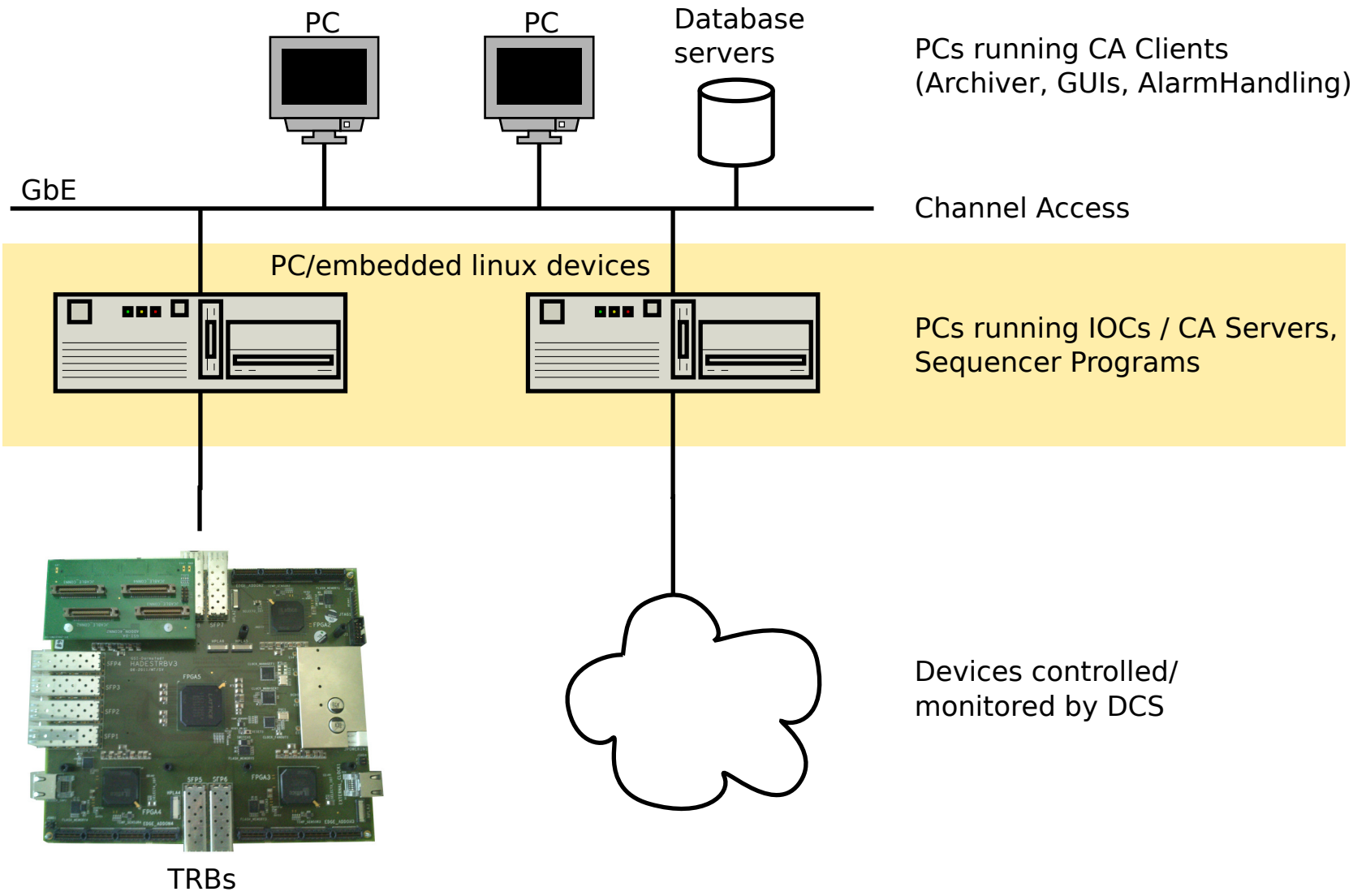
PCs running IOCs / CA Servers,
Sequencer Programs

Devices controlled/
monitored by DCS

Example: PANDA LMD



Canonical Form of an EPICS Control System

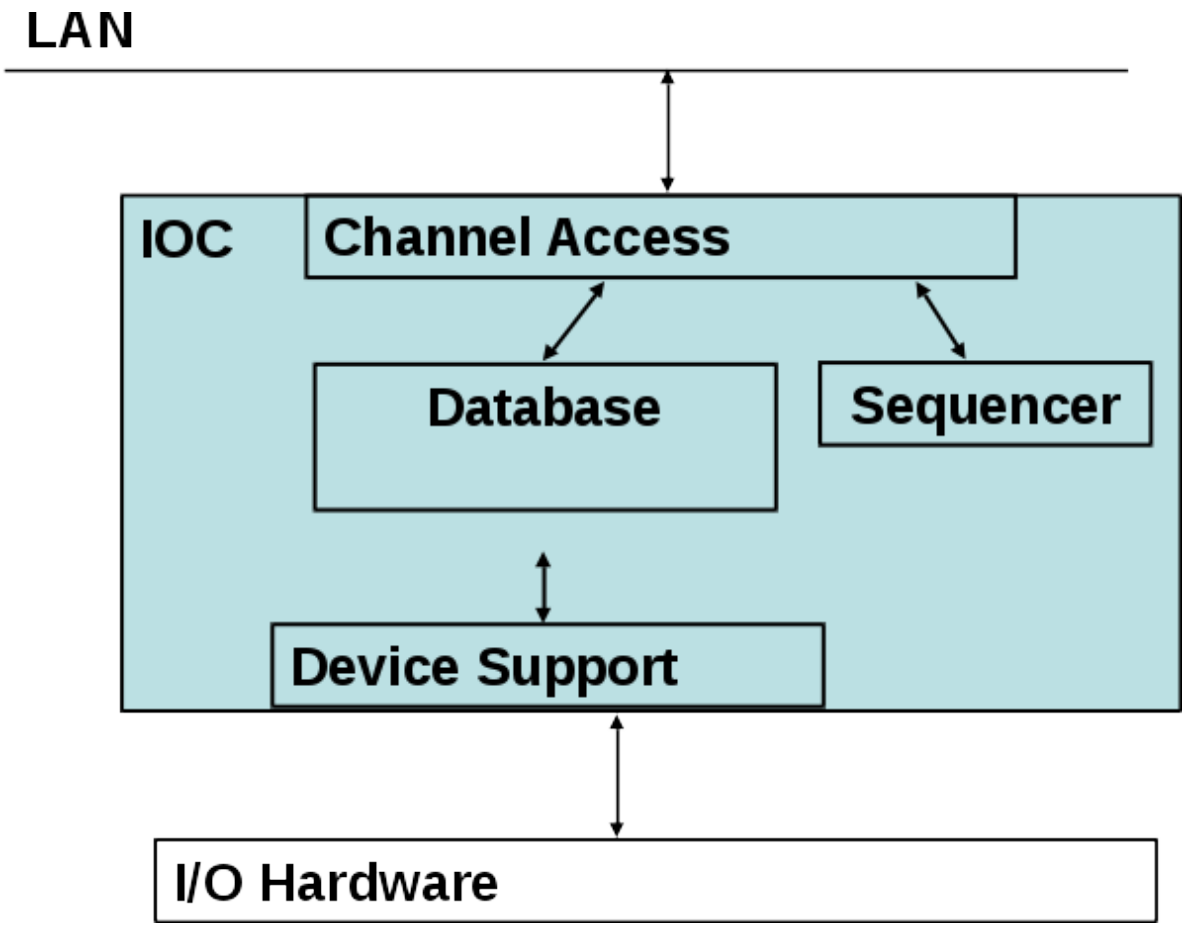


Key Feautures of IOC software

- Two primary application specific components:
  - The real-time database of records (required)
  - State Notation Language programs used to implement state oriented programs (finite-state machine, optional)
- Machine status, information and control parameters are defined as *records* in the application specific *database*.

Inside an IOC

The major software components of an IOC (IOC Core)



EPICS Records

- A record is an object with...
  - A **unique** name
    e.g. "PANDA:LMD:MUPIX:TEMP:H1:P2:M0:S1:D0:2:Tmom"
    (Temperature of MuPix sensor H1:P2:M0:S1:D0:2)
  - Controllable properties (fields) e.g. "EGU" (Engineering Unit)
  - A behavior - defined by its record type
  - Optional associated hardware I/O ($\Rightarrow$ device support)
  - Links to other records
- Records have some functionality associated with them (scaling, filtering, alarm detection, calculations, etc). Different record types have different functions and uses.
- *A record does nothing until it is processed!*
- Records can be processed periodically or event driven (Hardware interrupt, CAput, ...)

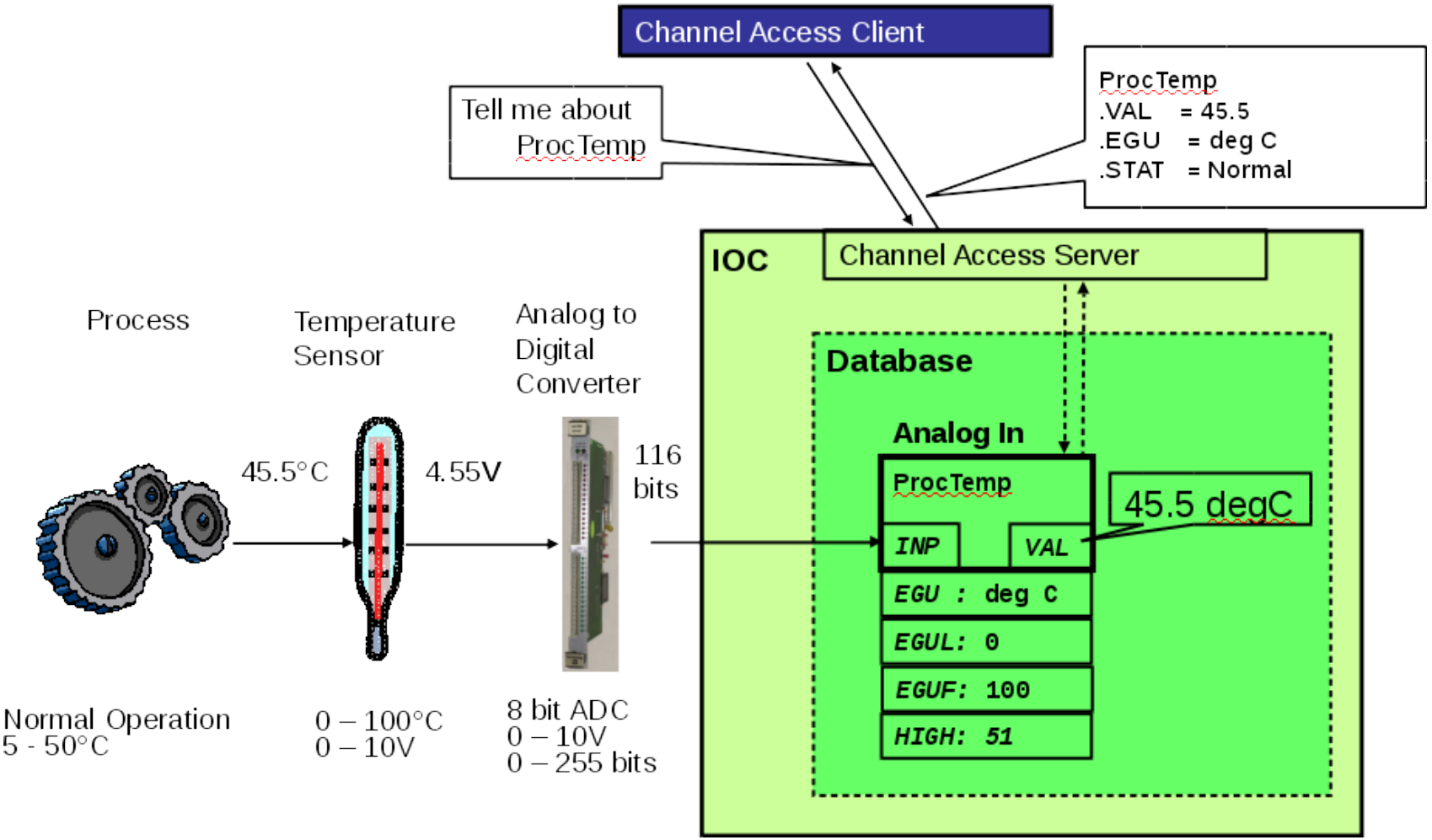
EPICS Records II

- Each field can be accessed individually by name
- A record name and field name combined give the name of a *process variable (PV)*
  e.g. "PANDA:LMD:MUPIX:TEMP:H1:P2:M0:S1:D0:2:Tmom.EGU"
- PV is named piece of data (with attributes)
- Examples for attributes: timestamp, status, alarm severity, ...
- CA uses PV names to access data

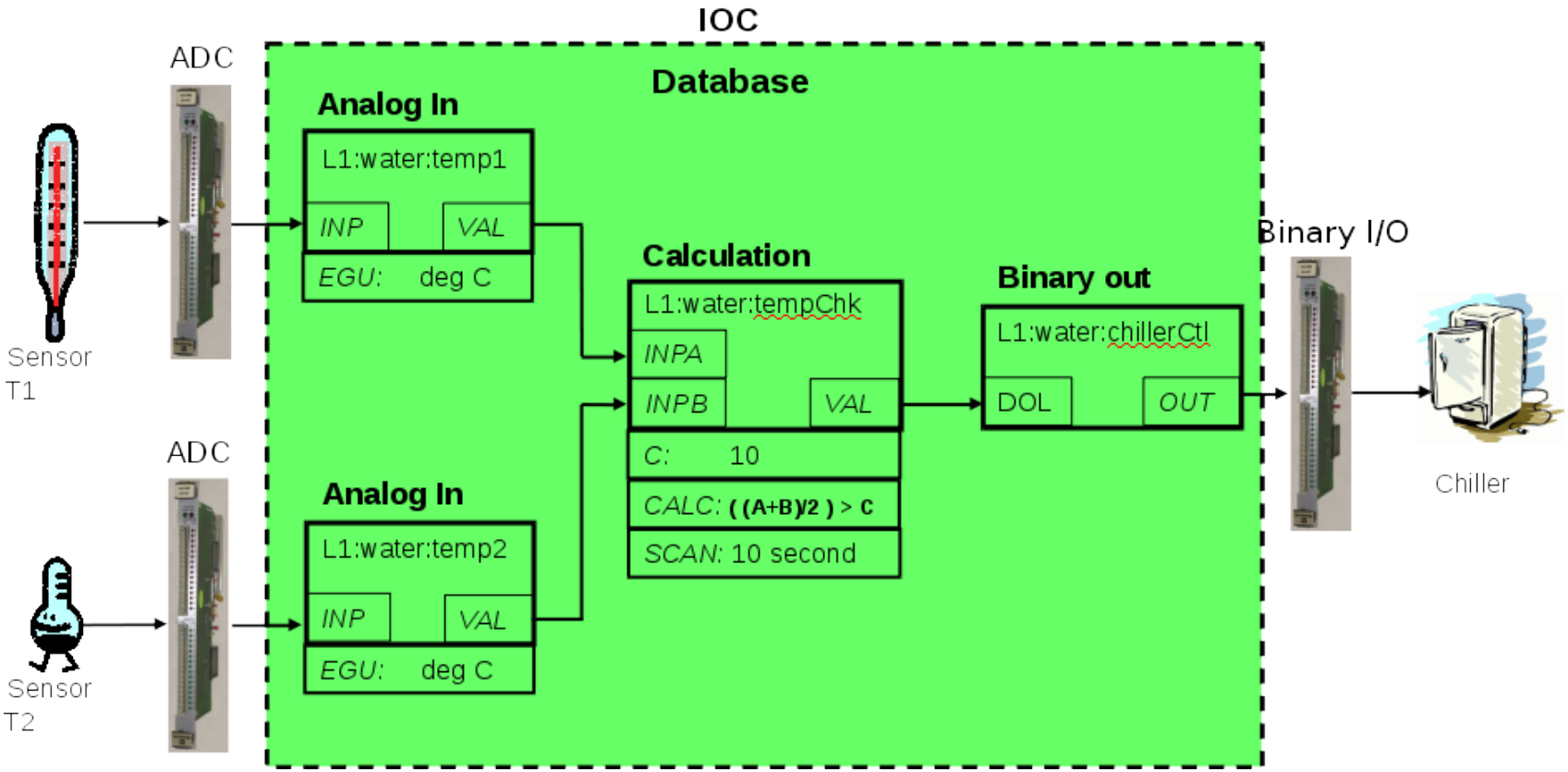
EPICS Database

- A collection of one or more EPICS records of various types
- Records can be interconnected and are used as building blocks to create applications
- A data file that's loaded into IOC memory at boot time
- Channel access talks to IOC's memory copy of database

A Simple Database

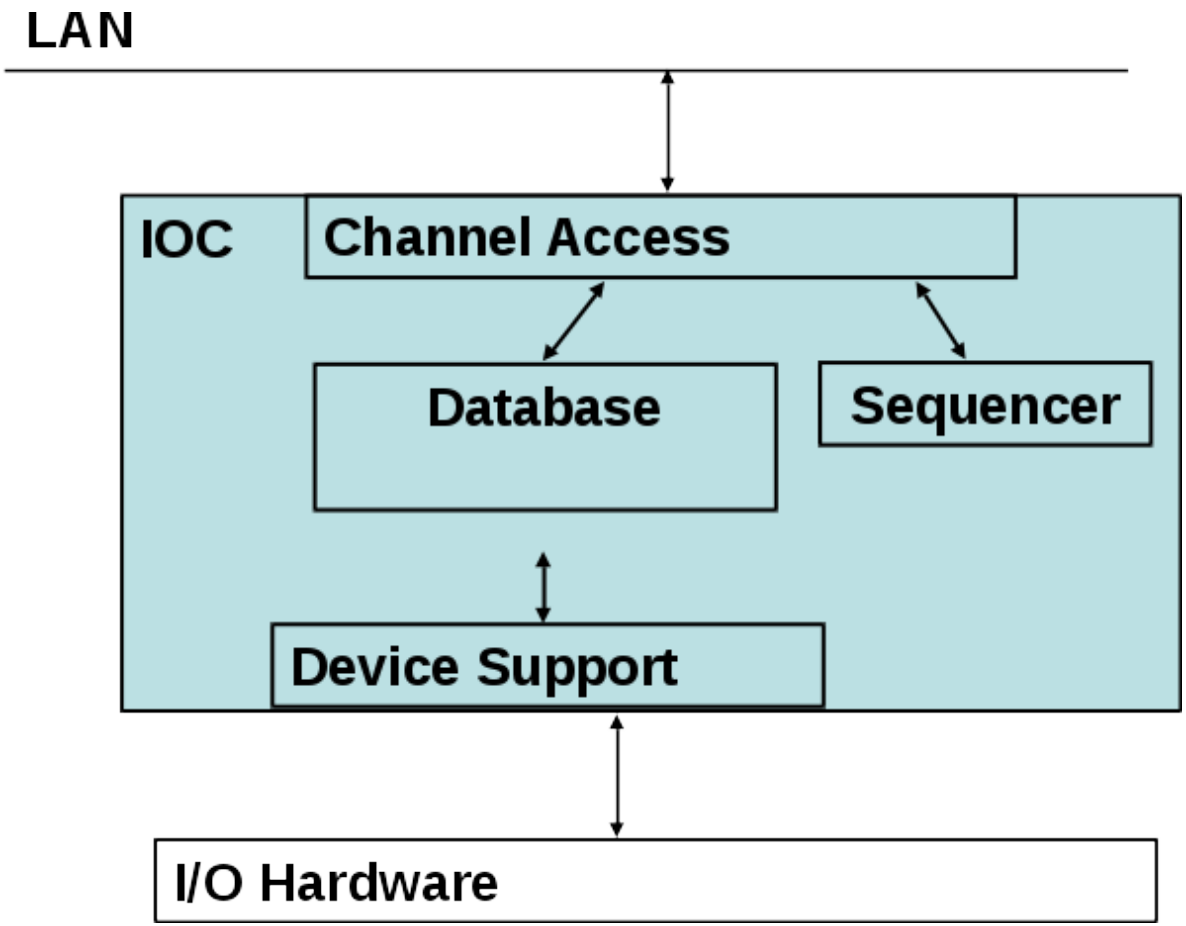


Database Processing



Inside an IOC

The major software components of an IOC (IOC Core)

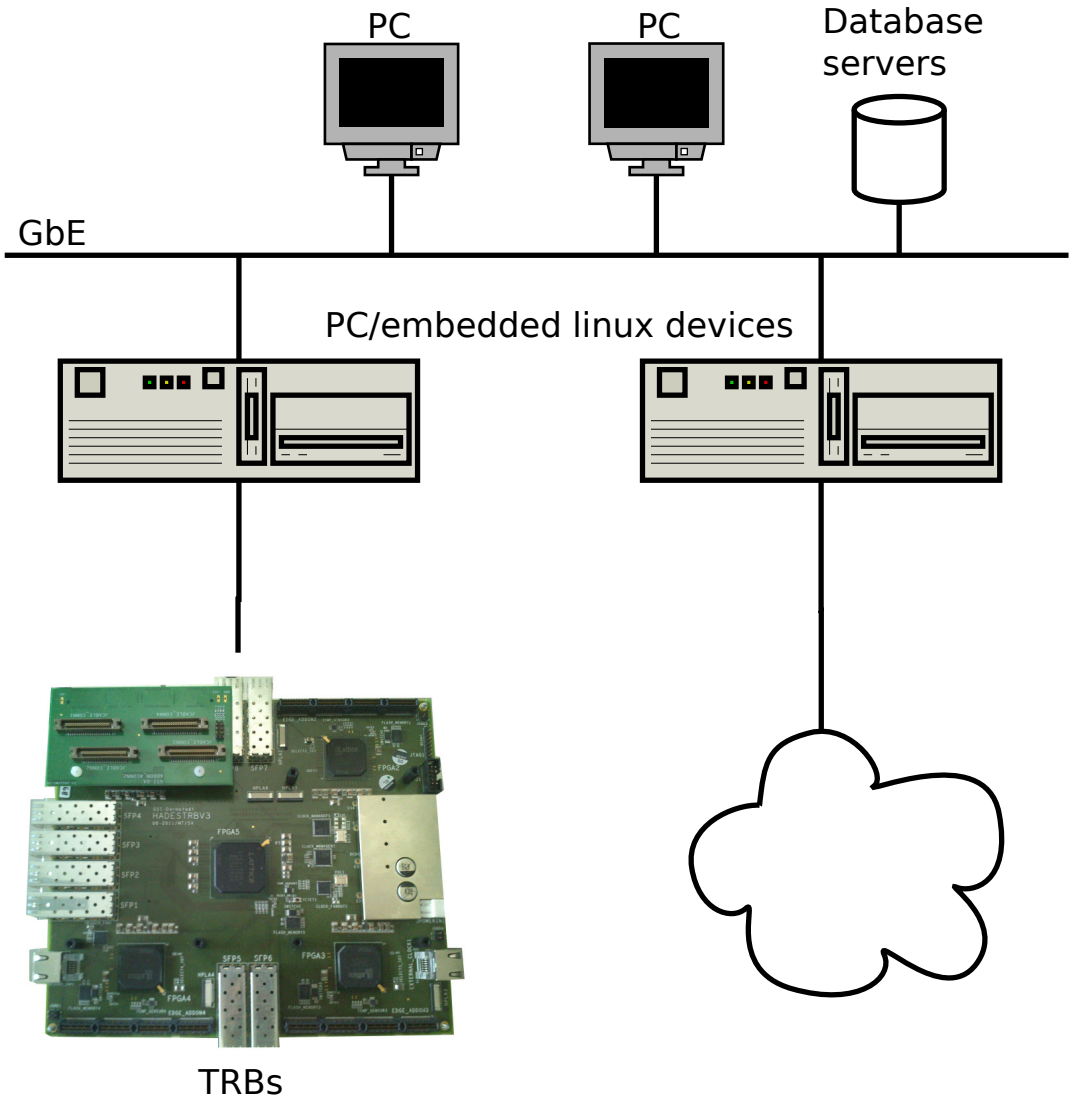


Device support

- Defines hardware I/O for a record type
- Simple *C* interface:
  init (global, per record instance), read/write, I/O interrupt handling
- Functions need to be implemented for each record type used
- Device Support uses underlaying API to hardware
  $\Rightarrow$ libtrbnet/DIM

- Device Support for DIM available:
  `https://github.com/zumbruch/EPICS-DIM`
  written by Peter Zumbruch (GSI)
  EPICS IOC can act as DIM server and DIM client

Canonical Form of an EPICS Control System



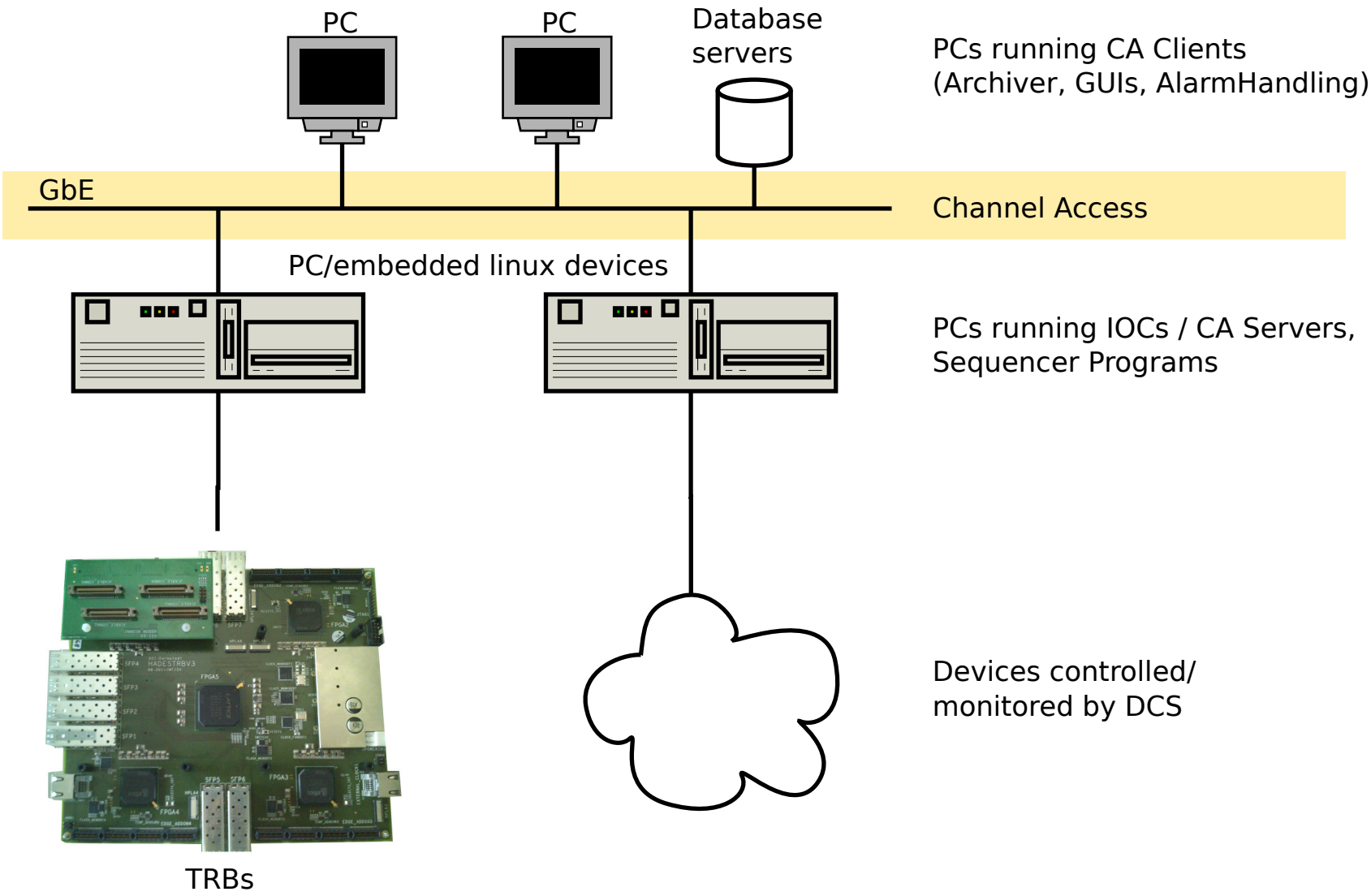
PCs running CA Clients
(Archiver, GUIs, AlarmHandling)

Channel Access

PCs running IOCs / CA Servers,
Sequencer Programs

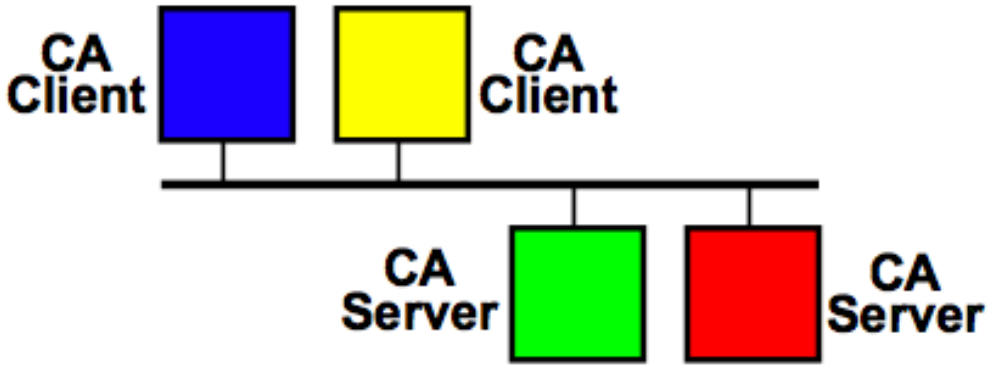
Devices controlled/
monitored by DCS

Canonical Form of an EPICS Control System



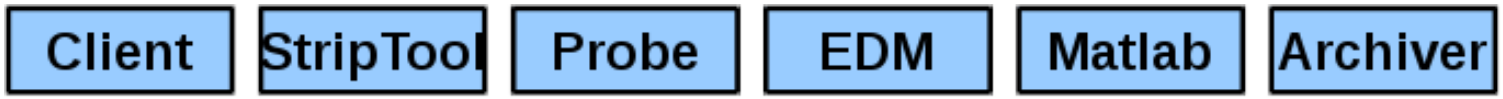
Channel Access Protocol

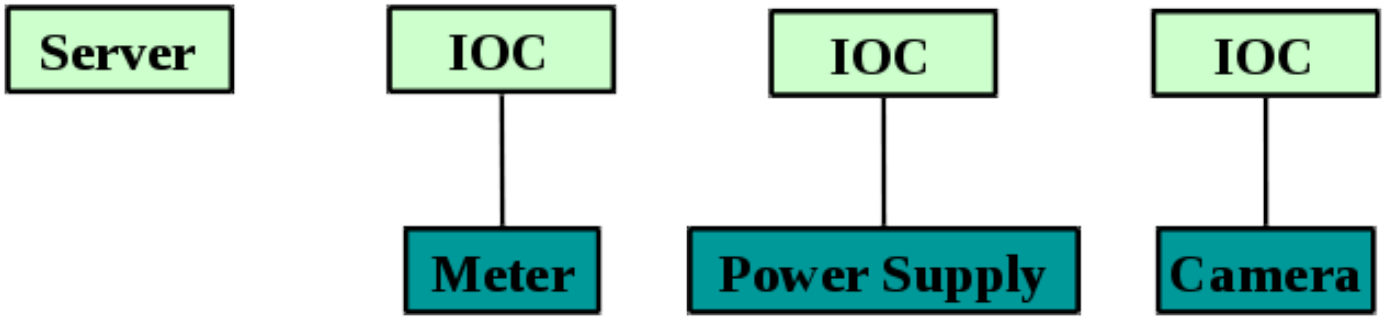
- Network-based "client/server" model
  Also known as "publish/subscribe"



- Channel Access (CA) is the protocol that connects clients and servers (virtual bus)
  - Very efficient (high troughput, low latency, no polling)
  - Robust (partial degradation, incremental recovery)
  - Self-configuring (uses "discovery" protocol)
  - Scalable (100s of elements, 100000s of connections)
  - Simple API: "set", "get", "monitor"

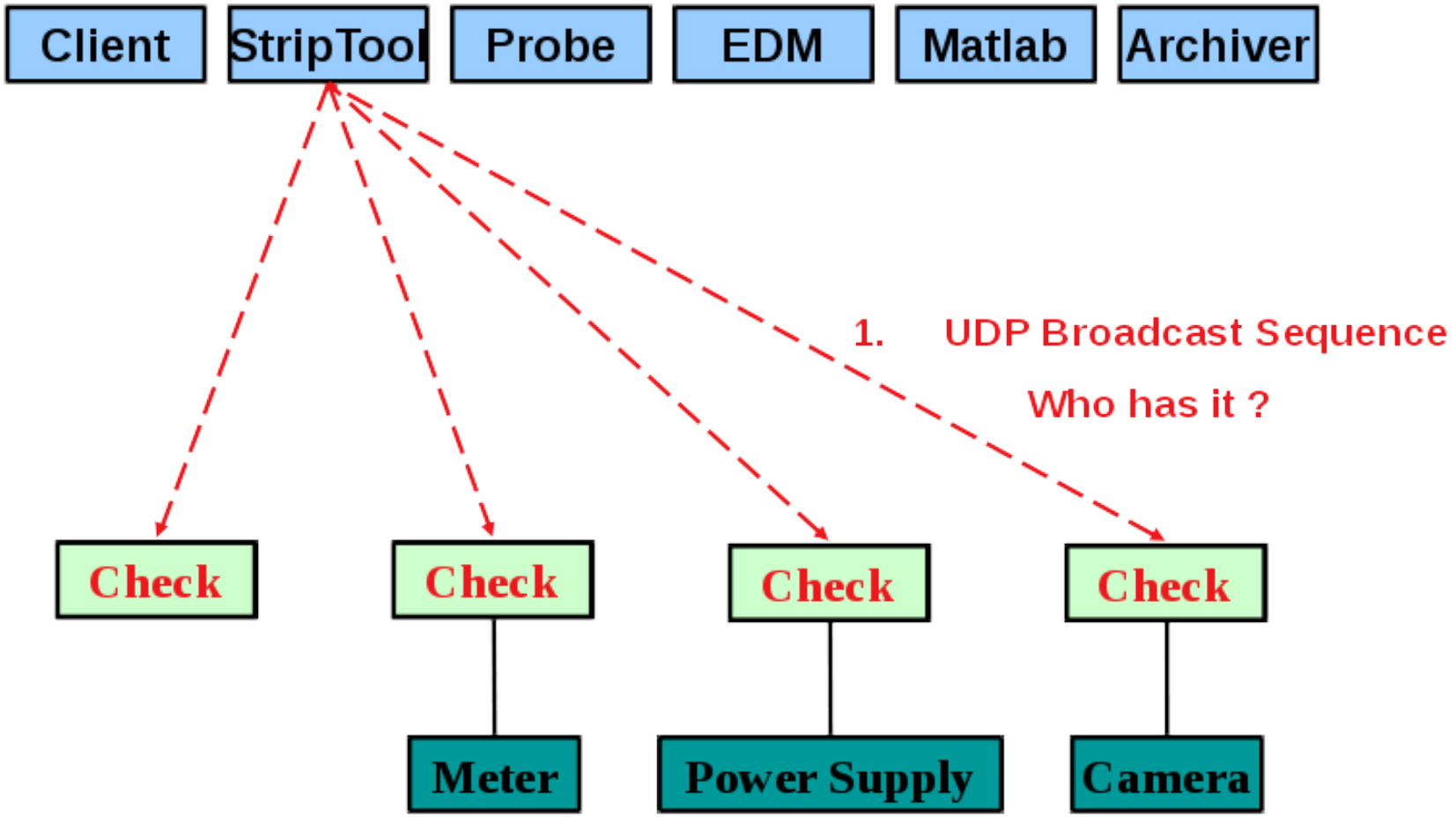
CA Search and Connect Procedure





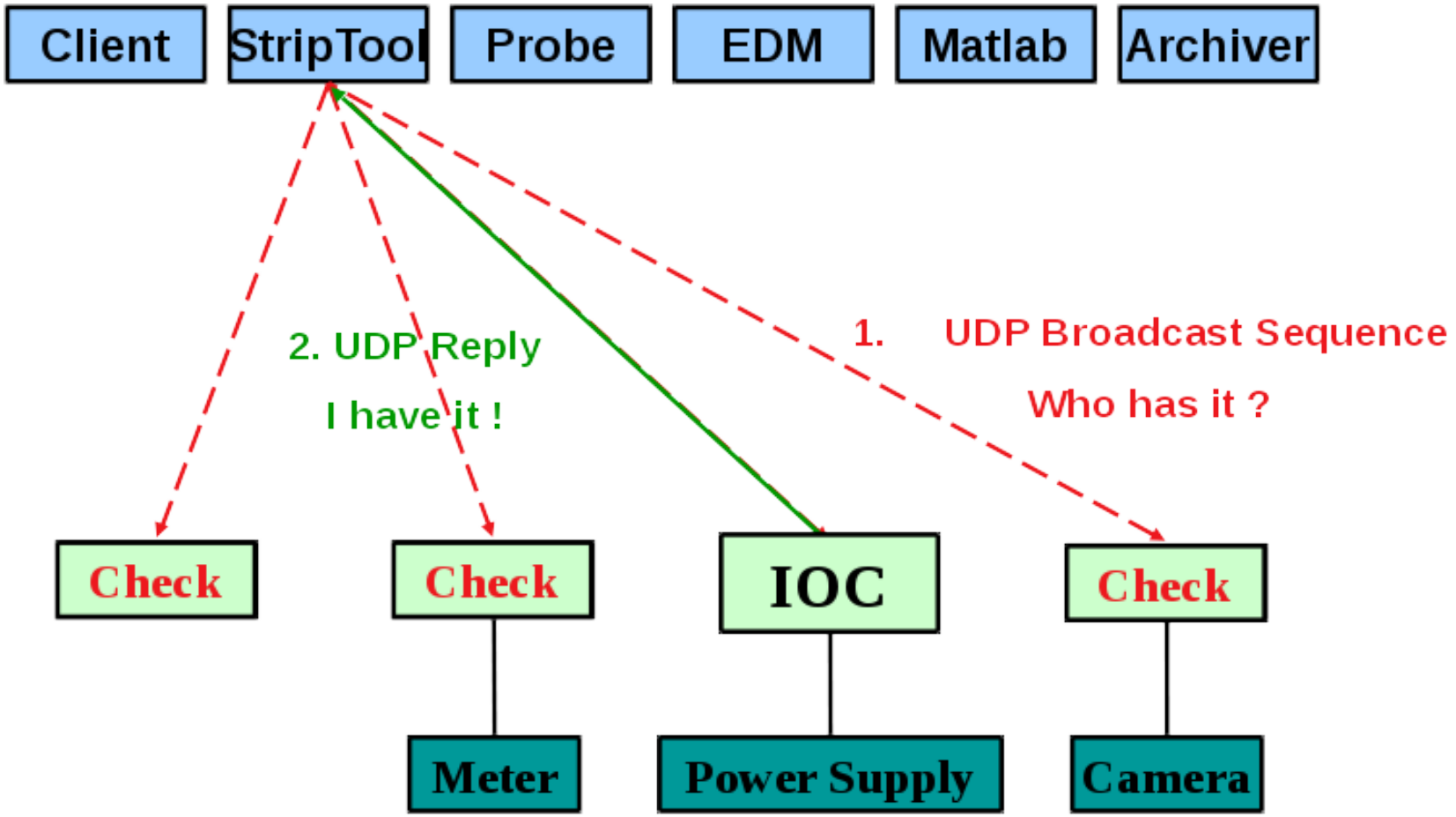
Based On Getting Started with EPICS Lecture Series "Introduction to Channel Access Clients"
Kenneth Evans, Jr.

CA Search and Connect Procedure



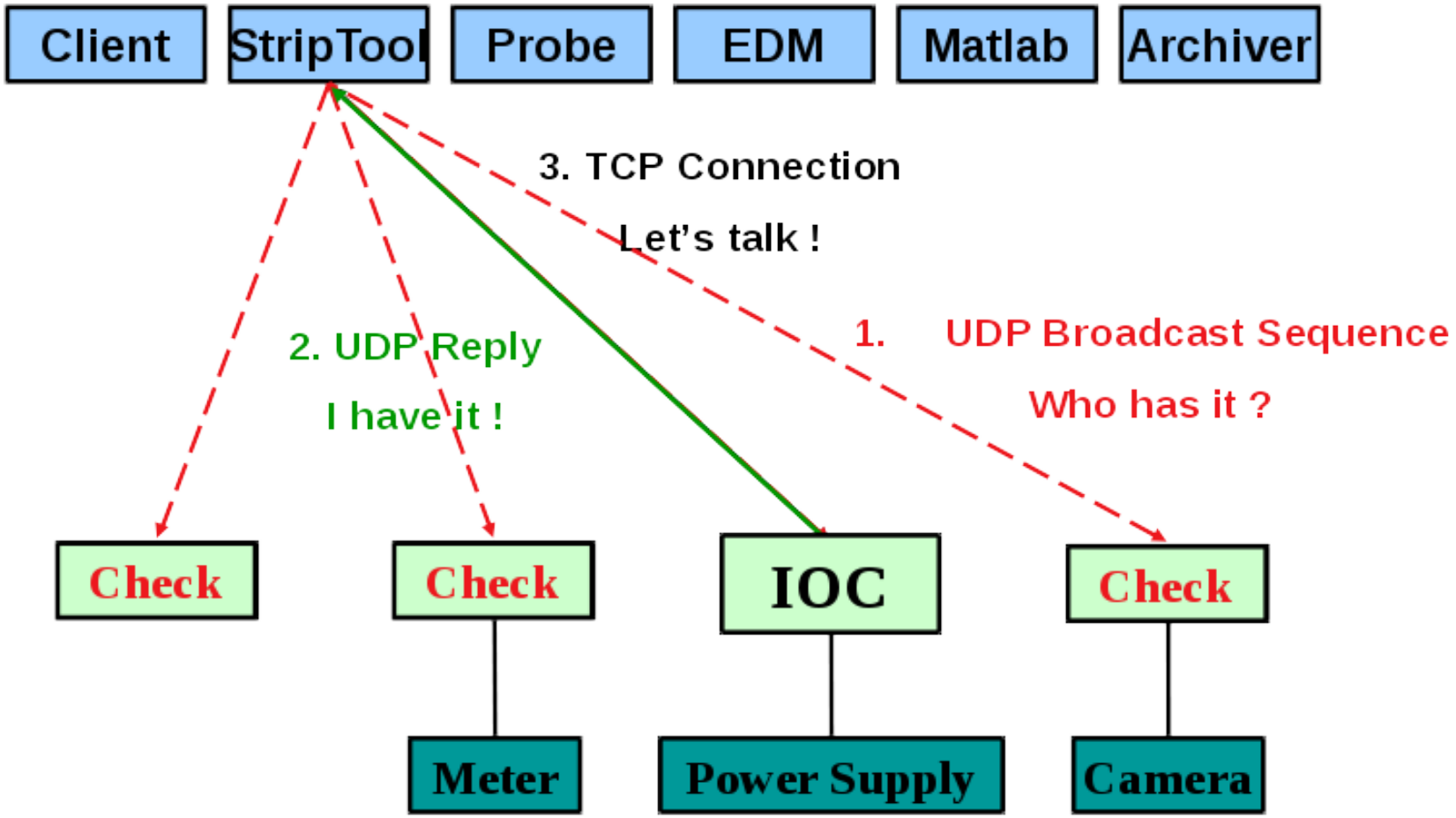
Based On Getting Started with EPICS Lecture Series "Introduction to Channel Access Clients"
Kenneth Evans, Jr.

CA Search and Connect Procedure



Based On Getting Started with EPICS Lecture Series "Introduction to Channel Access Clients"
Kenneth Evans, Jr.

CA Search and Connect Procedure



Based On Getting Started with EPICS Lecture Series "Introduction to Channel Access Clients"
Kenneth Evans, Jr.

Channel Access Client

- Alternative method to connect TRBnet/DIM with EPICS:
- Write stand-alone program for communication with TRB
- Interaction with EPICS via Channel Access (CA)
- CA implementation available for different programming languages: C, Perl, Python, Tck/TI, Java, ...
- CA is very efficient with a simple API
  BUT! Bad programming can easily kill network due to high traffic (overhead)

Conclusion

- Two possibilities to let EPICS talk with TRB:
  Device Support or Channel Access Client
- Need to think about requirements and goals!
- What do we want to do on the TRB/Front ends using EPICS?
  - Check Status of front end?
  - Monitor supply voltages?
  - Check firmware version?
  - Configure front end?
  - . . . ?