ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-THCA5

# RETHINKING PLCs: INDUSTRIAL ETHERNET FOR LARGE-SCALE REAL-TIME DISTRIBUTED CONTROL APPLICATIONS

B. Ploetzeneder, O. Janda, A. Krucenko, J. Trdlicka, P. Pivonka, P. Bastl
ELI Beamlines/Institute of Physics of the ASCR, Prague, Czech Republic

*Abstract*

Many research facilities rely on PLCs to automate large slow systems like vacuum or HVAC, where price, availability and reliability matter. The dominant architecture consists of local units of controllers/modules (programmed in IEC61131-3 languages), which operate mostly autonomously from a SCADA layer.

While some vendors provide low-level stacks to encourage growth of their ecosystems, PLC programming remains largely within a closed, proprietary world.

In this paper, we introduce a different way of thinking about PLC hardware.

Working with the open stacks intended for the design of new EtherCAT (Beckhoff)/Powerlink (B&R) modules, we built an abstract C++- API to control the existing ones. These industrial Ethernet busses can be propagated using standard network hardware, so any RT-Linux system can now control any PLC module from anywhere in our facility using high-level languages (C++, LabVIEW).

This way, PLC modules are seamlessly integrated into our distributed TANGO-based control system. PC-PLC interfaces are no longer needed; or in the case of traditionally implemented subsystems, trivial.

## BACKGROUND

Programmable logic controllers (PLCs) are cheap, reliable, modular and fault-tolerant "hard" real time control solutions, even in challenging environments. They are easy to program and maintain, and allow online modification of hardware/software.

With a few notable exceptions [1], most facilities rely on a standard architecture shown in Figure 1: Local units (processors and modules) are distributed in the field, and operate almost autonomously from a SCADA layer, interacting using thin communication links (Profibus, Modbus, OPC, custom serial/Ethernet interfaces, ...).

This approach has a number of disadvantages:

- It requires control system engineers with skillsets that are not commonly taught together (IEC61131-3 languages for the PLCs vs. high-level languages for the SCADA systems).

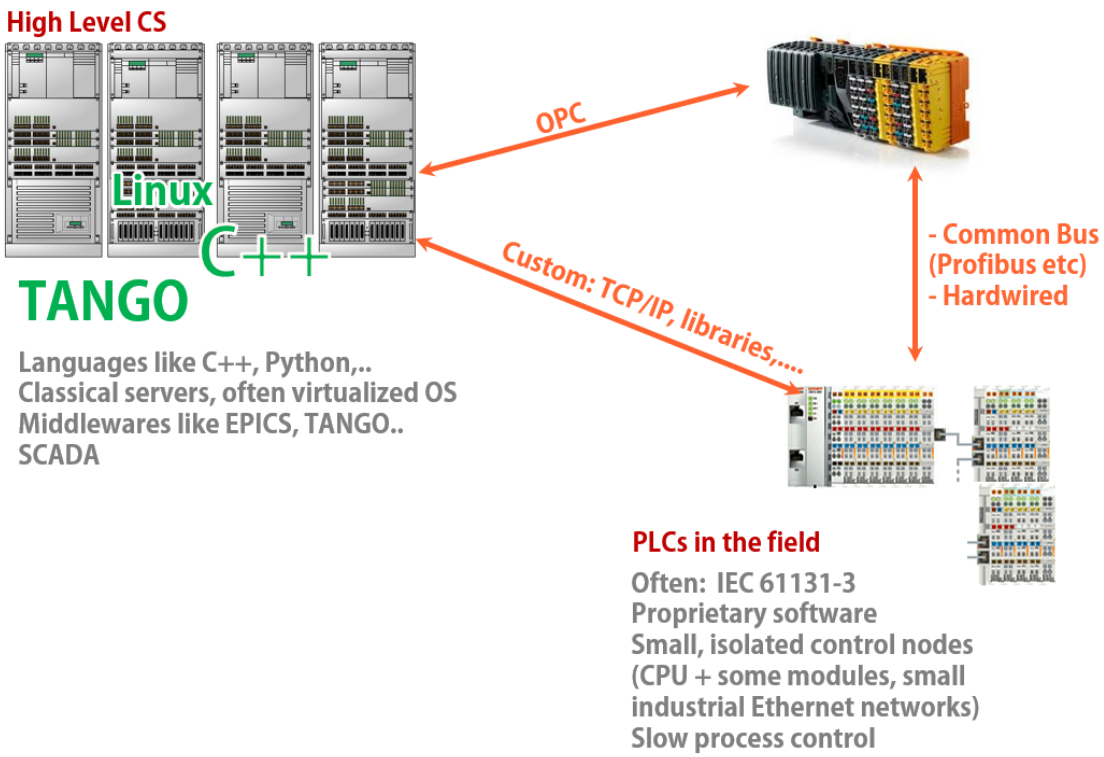


Figure 1: Commonly found PLC architectures.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-THCA5

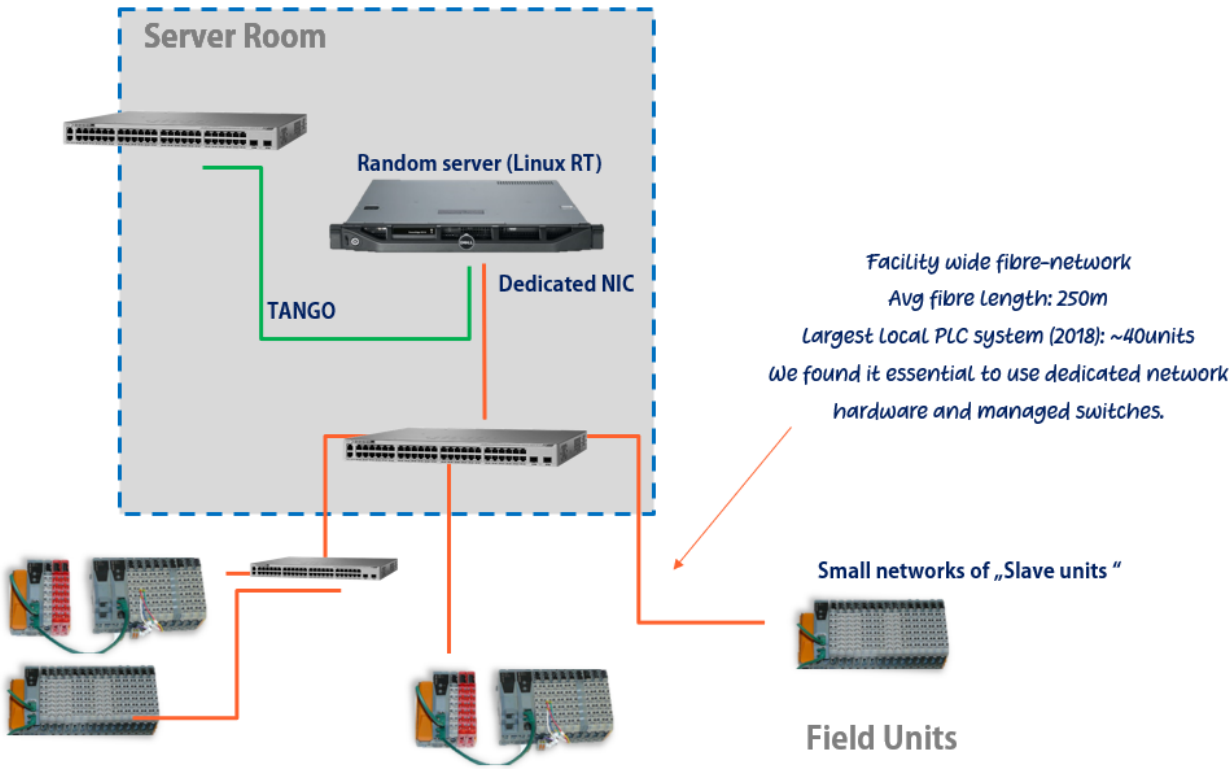


Figure 2: Alternative concept for propagating Industrial Ethernet systems.

- It requires the development and maintenance of software development infrastructure for both PLCs and SCADA systems.
- Interfaces are costly in terms of resources (require manpower, customized solutions).
- There is limited flexibility and the system is not truly distributed.

## ALTERNATIVE CONCEPT

While PLC programming mostly remains within a closed, proprietary world, a few vendors provide low-level stacks to encourage growth of their ecosystems, most notably Beckhoff with EtherCAT [2] and B&R with PowerLink [3]. Both companies formed some form of consortium to make their standard freely available, and provide documentation and even software implementations for developers of new modules (slave units).

We tested and implemented prototypes for both technologies, and while we prefer some of Beckhoffs' configuration possibilities and the design of their IPCs, B&R has the more attractive selection of modules, the "reaction" technology and modules that can be directly connected to fibre-based networks.

The availability of those stacks including software implementations, however, also allows the replacement of the master units. This means, you can remove the CPUs which drive the modules using proprietary software, propagate the field busses via standard networks (copper, fibre) and drive them with custom software (see Fig. 2).

The hardware can be then exposed facility-wide via the middleware and allows truly distributed access.

In our experience, it is helpful to use dedicated network hardware and managed switches for this purpose. It is possible to operate smaller, local networks over VLANs using cheap switches, but for larger systems and longer distances, we could not maintain real-time communication. Note that fibre-length in the hundreds of meters can limit cycle time due to signal propagation times.

This architecture has multiple advantages:

- The PLC hardware is seamlessly integrated into the high-level controls architecture.
- CS Engineers can use high-level languages like C++ to drive the hardware.
- Interfaces to "classically programmed" PLCs (for example, from external contractors or for safety-critical systems) are trivial using communication modules like Profibus.
- The solution can significantly reduce cost-per-channel by removing expensive CPU units.
- Only the slave units are deployed in the field and exposed to EMP, radiation, … the driving hardware is safely located inside the server room. This reduces maintenance cost and increases availability.
- The approach is highly flexible; with our software, adding channels to the control system requires no programming, just configuration and a bus restart.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-THCA5



## RESULTING SOFTWARE

To implement this system, we provide two APIs:

- **Fieldbus:** An abstraction of the stacks that drive the industrial Ethernet networks (Beckhoff and B&R), simplifying bus management to a few commands (start the bus, stop the bus, module discovery..).
- **CSIO:** This abstract class (with specific implementations like 'DIO', 'AIO', 'Stepper') is used to access the individual modules.

To use PLCs, a user has to go through the following steps:

1. Physical setup including network infrastructure
2. Configuration of the topology and cycles in Automation Studio/System Manager. This step can technically be also done by custom implemented software, but we currently don't find value in doing so.
3. Starting the bus, for example using a dedicated TANGO Server.
4. Discovering modules via the Fieldbus API. When calling this method, a collection of CSIO objects is returned. Our TANGO Server currently uses this to create dynamic attributes for access; though later on we wish to expose the module via individual servers.
5. Working with the modules. The methods of each CSIO implementation correspond to some register-read/write command. After each call, the bus has to be "synced", corresponding to the typical cycles in a PLCs. This is hidden by our TANGO server.

This set of APIs is also the base for a "native" LabVIEW integration [4], which allows seamless use of B&R and Beckhoff hardware within NIs' LabVIEW

## OUTLOOK

The system shown is currently an operational protoype, running two beamlines and various small subsystems in ELI Beamlines. In the next months, we will

- Work on the upper abstraction layers and completely integrate the system into the local control infrastructure described in [5]. Specifically, we wish to hide the implementation specifics and use the same abstract API we use for any other motor/DIO/...: A user shouldn't know they are dealing with a fieldbus-based implementation. This also means exposing the modules as individual TANGO servers rather than as TANGO attributes.
- Quantify and optimize the performance and limitations of the drivers.

## REFERENCES

[1] D. Maier-Manojlovic, "Real-Time EtherCAT Driver for EPICS and Embedded Linux at Paul Scherrer Institute (PSI)", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 153–156. doi:10.18429/JACoW-ICALEPCS2015-MOPGF027

[2] EtherCAT, https://www.ethercat.org/default.htm

[3] Powerlink, https://www.ethernet-powerlink.org

[4] Birgit Ploetzeneder, "Native LabVIEW pro průmyslovou automatize", *TA ČR GAMA Grant*, 01/17-12/2019.

[5] P. Bastl *et al.*, "Hardware Architecture of the ELI Beamlines Control and DAQ System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 2000–2006. doi:10.18429/JACoW-ICALEPCS2017-FRAPL05