Open control systems : architecture, problems and solutions

Peter Wurmsdobler
Open Control Laboratory
Control.com
peter@control.com

Outline

- Control systems
- Open control systems
- Open control architecture
- Problems and solutions
- Example: PLC
- Conclusion

Different perception of 'control'

Feed-forward and Feed-back control, or

- Motion, robot and machine control
- Plant, batch and process control
- Distributed and supervisory control

or Fuzzy, Adaptive or Predictive control ?

Control system implementation

Depending on speed, complexity and level :

- Micro-controller, μC, to control motion
- Programmable Logic Controller, PLC, to control a machine
- PC-based controller, to implement supervisory control

Control system contains cascaded loops with complex system behavior

Control system requirements

- Deterministic,
- Predictable,
- Rugged,
- Scalable,
- Redundant,
- Fail-safe,
- Maintainable.

Definition of term 'open'

Different interpretations can been heard, such as

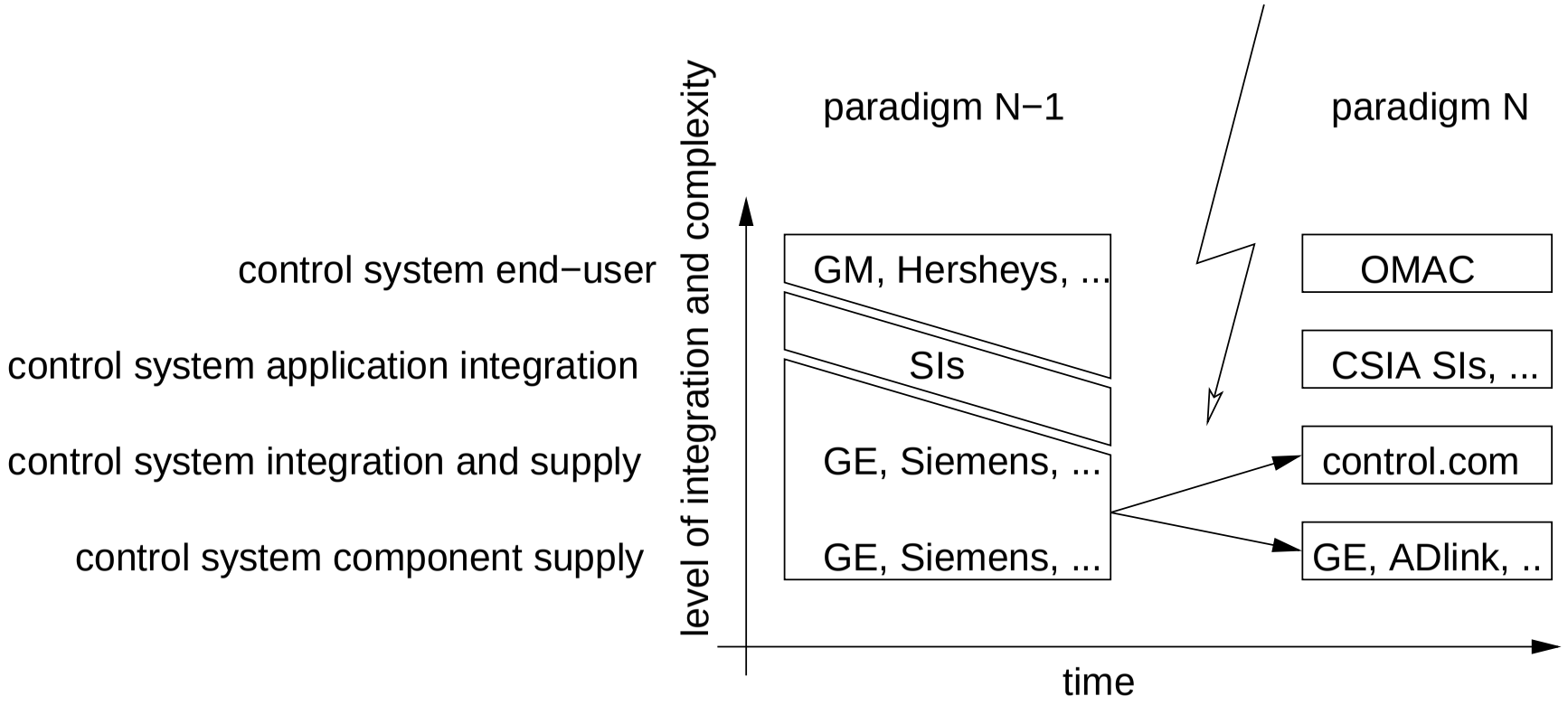
Open = widely used corporate technology,

Open = well documented corporate technology,

Open = not patented technology.

"An open system is composed of components with interfaces complying to vendor independent standards"

Open control paradigm shift



The role of control.com and the OC Lab

- Establish an open forum of discussion
- Establish an infrastructure to meet user needs
- Establish a laboratory for
  - working on open control architecture and interfaces
  - testing for interoperability between components
  - measuring performance of components
- Promote the open control idea

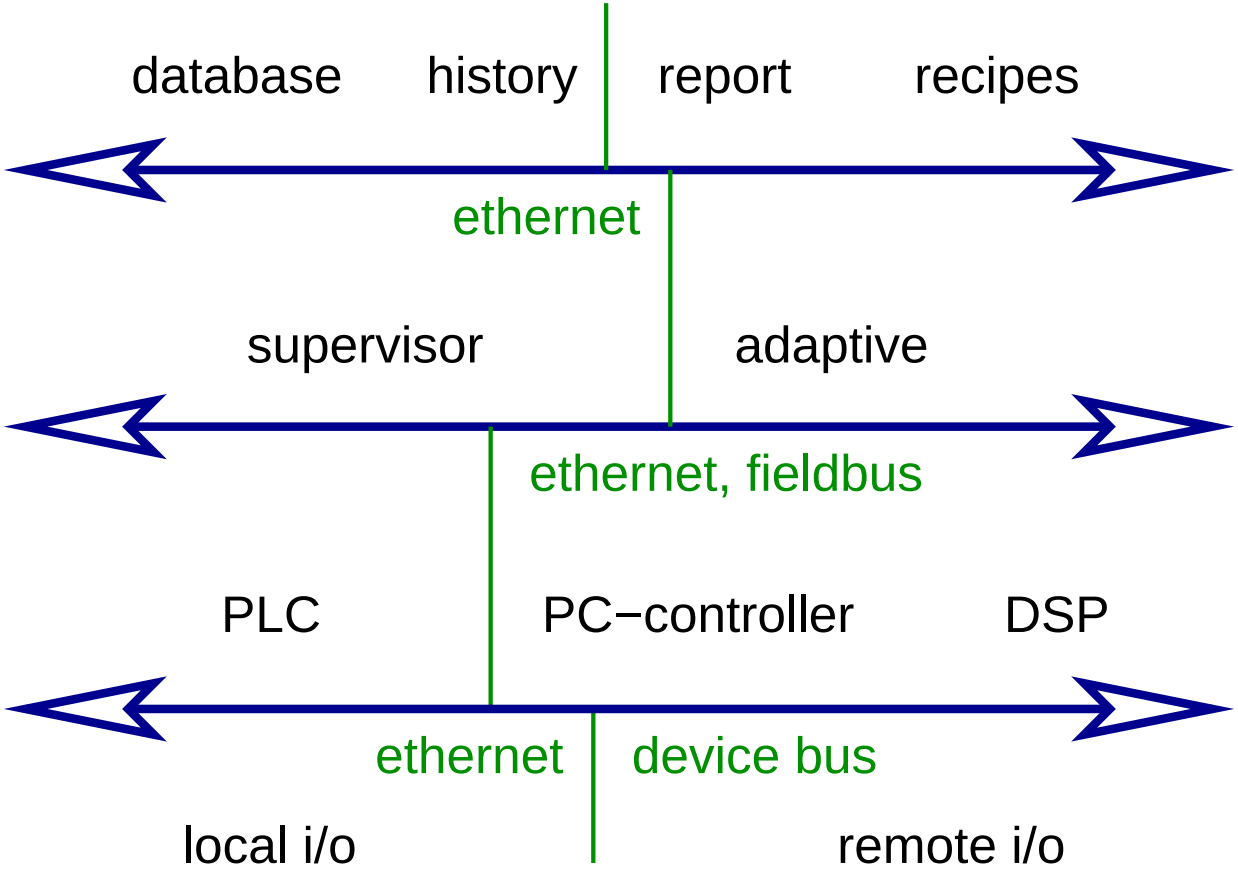
Open control paradigm advantages

- Components
- Modular
- Extensible
- Portable
- Scalable
- Maintainable
- Independence

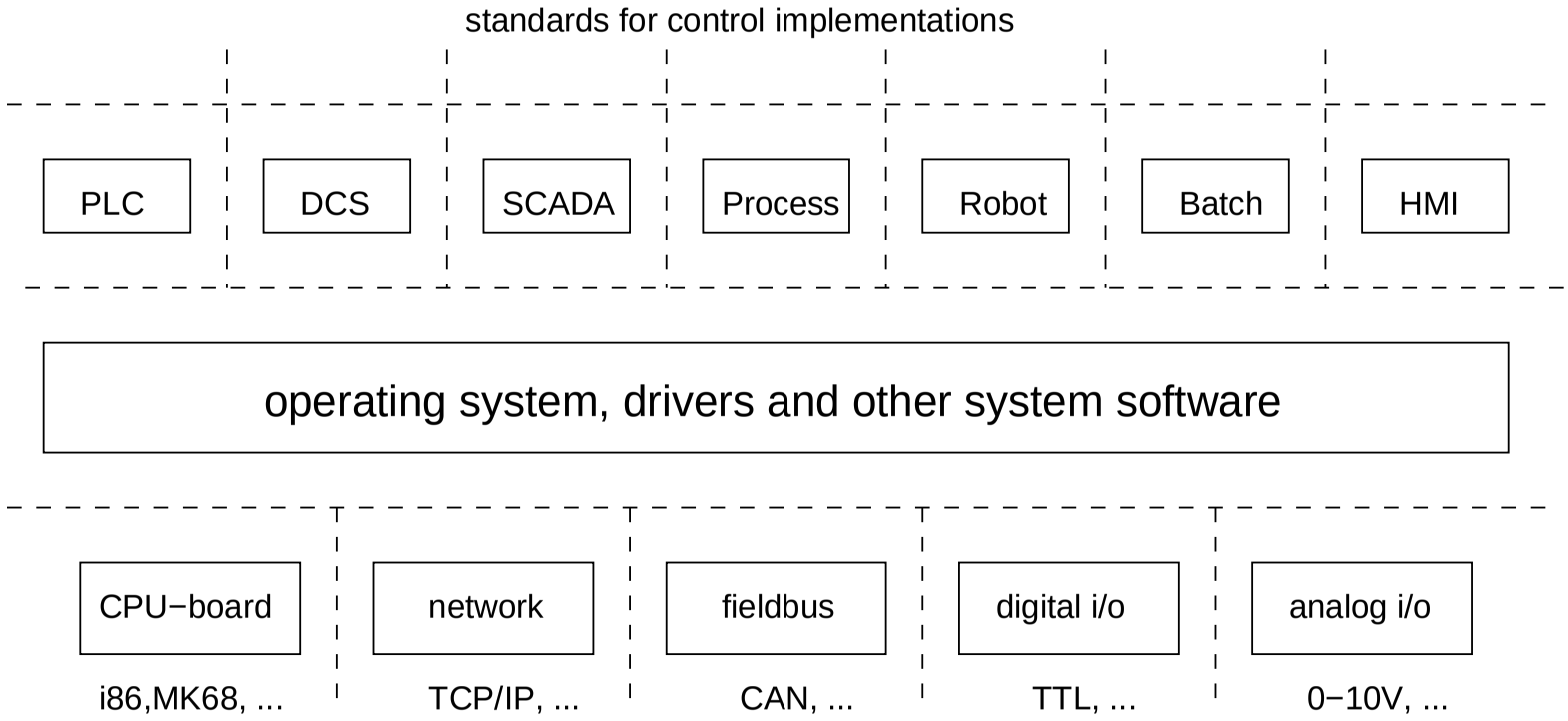
Open control system efforts

- LinuxPLC, an open source effort for PLC on Linux
- PuffinSCADA, an open source effort for SCADA on UNIX
- MCA, Modular control architecture
- ORCOS, open robot control systems
- OMAC, open modular architecture controller
- OSACA, European open control group
- JOP, Japanese open control

Control enterprise architecture



Controller architecture



Problems of open architecture

Dotted lines represent interfaces, hence:

- Establish open interface standards
- Conformity to interface standards
- Interoperability of components
- Performance of components and system
- Real-time behavior of complex system

Interoperability

- HW: PCI, ISA and friends, e.g.: how many pci busses are in a compactPCI systems ?
- HW-SW: kernel and driver, e.g.: version numbering or can the driver handle more busses ?
- OS-API, e.g.: how POSIX is POSIX or why does send() react differently ?
- protocols, e.g.: is OPC reliable?

Real-time behavior

Closed loop control system susceptible for timeliness:

- OS schedules periodic control thread:
  timeliness uncertainty will generate noise
- Network i/o has unpredictible delay:
  unknown delay increases instability

However, distributed i/o and computing is more and more common.

Programmable Logic Controller

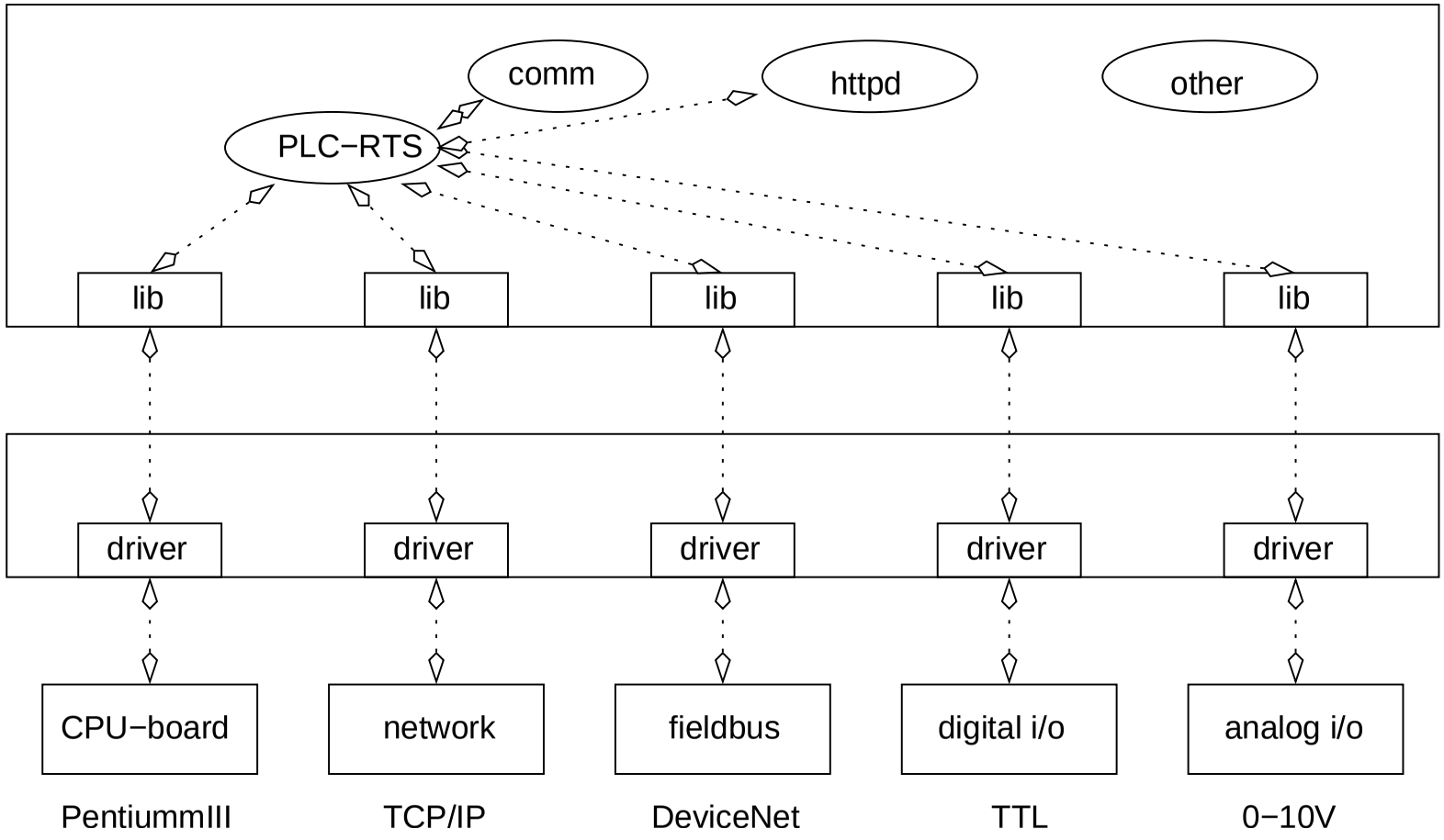
First, logic control was done by pneumatic or electric relays,

PLC emulates relays logic by "PLC paradigm", scan loop:

```
while(1)
    {
    read_inputs();
    do_control_logic();
    write_outputs();
    }
```

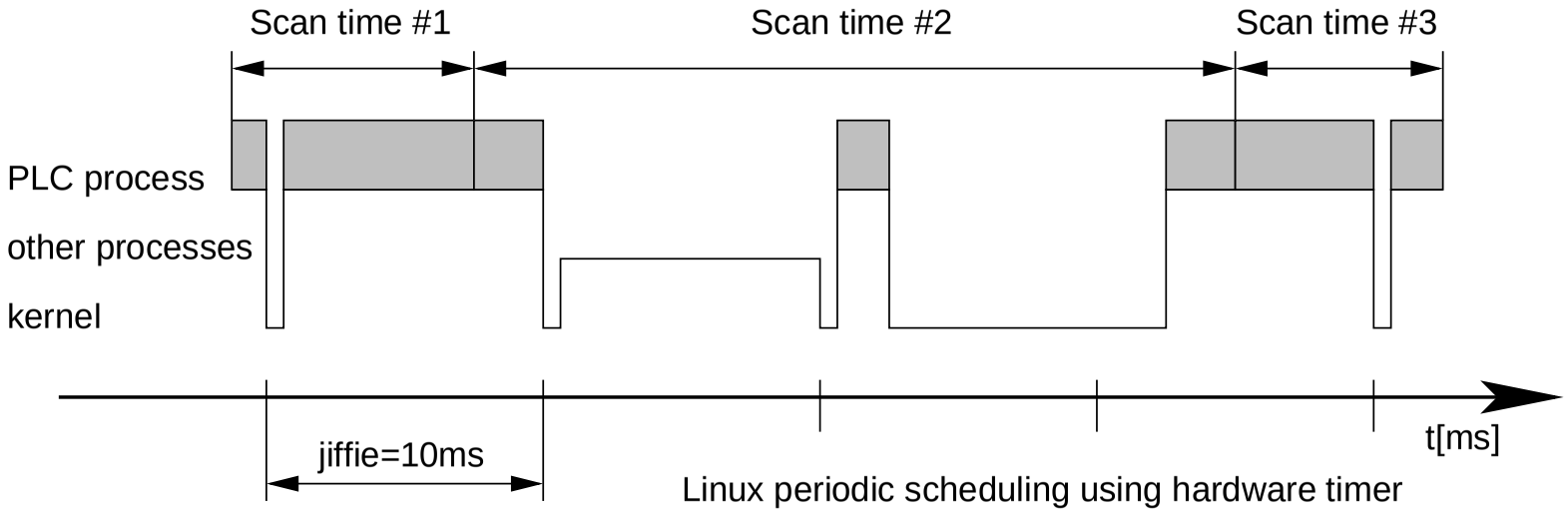
Logic is still represented in Ladder diagrams and relay logic.

SoftPLC : emulating a PLC on OS



Real-time performance I

Important performance indicator: PLC scan time



Real-time performance II

Time stamp difference between two gettimeofday():

```
less than       counts
        1 us  28238601
       10 us  71755198
      100 us  5701
     1000 us  332
    10000 us  44
   100000 us  101
  1000000 us  23
worst case = 593185 us
```

Local vs. remote i/o

Trend towards remote i/o and SoftPLC, but:

- Remote i/o not in polling, but asynchronous scans
- Multiple frequencies in a control system
- scan time period may fluctuate drastically on multiprocessing system

Requirements for OS in open control

- Run one process unfair, the rest fair
- Develop open interface standards
- Synchronize remote and local i/o processes
- Compensate for delays and timeliness uncertainty
- Portable APIs and open source code
- Volunteers in the Lab