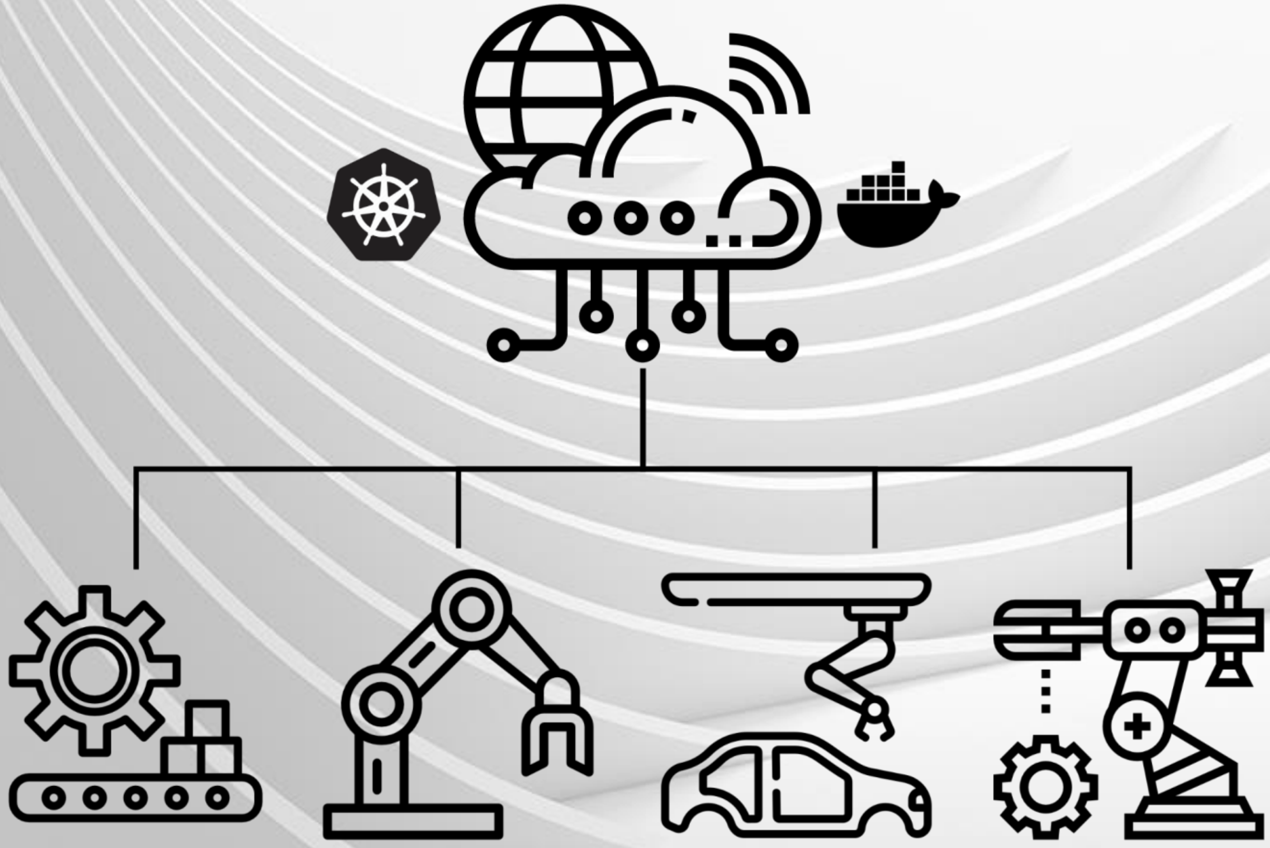
PUBLIC



MASTER THESIS

Updating Industrial Automation Software in Cloud-Native Environments

Shardul Sonar

01.05.2020 – 30.10.2020

Supervised By

Dr. Ing. Heiko Koziolek

Dr. Ing. Julius Rückert

Prof Christoph Hahn

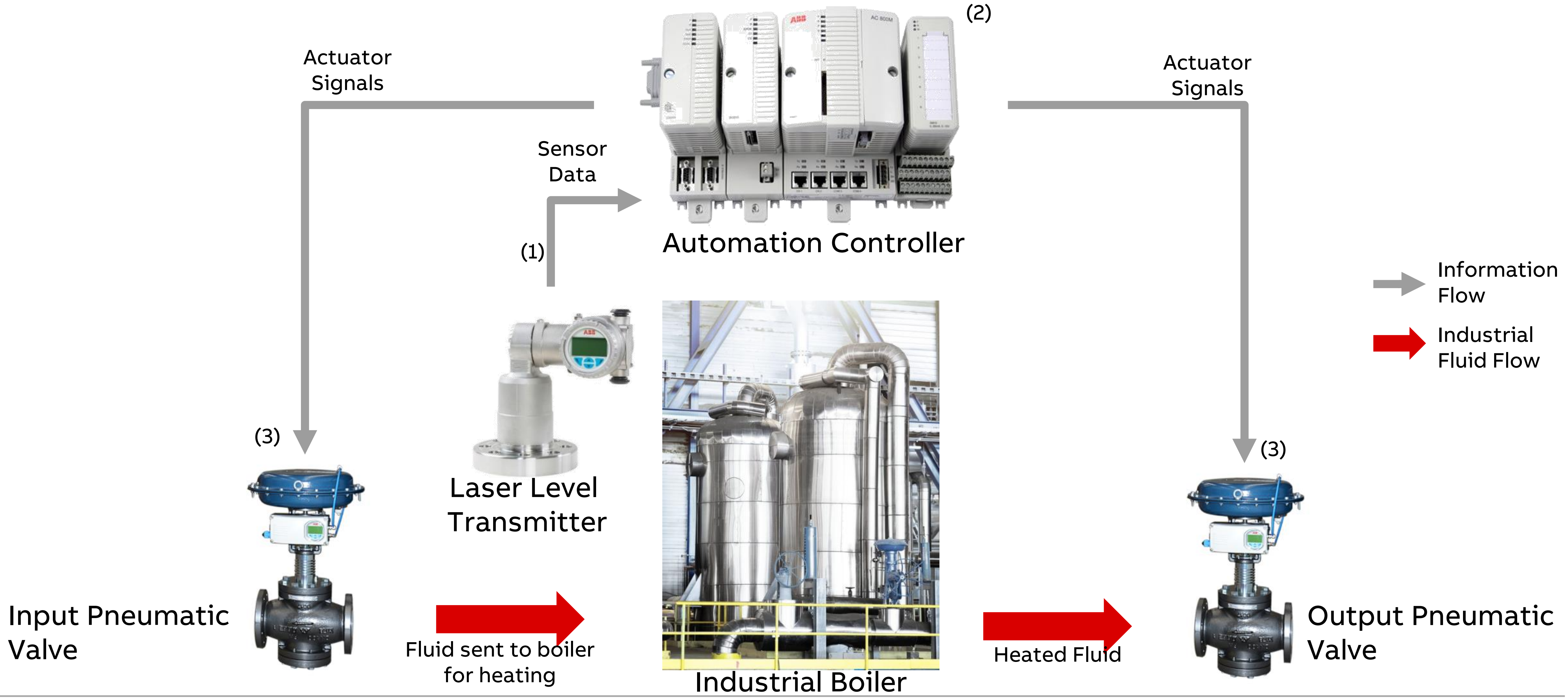
Prof Dr. Gerd Moeckel

HOCHSCHULE SRH HEIDELBERG Intelligence in Learning

ABB

1. Introduction

Industrial Control Application for Maintaining Fluid Level in Boiler

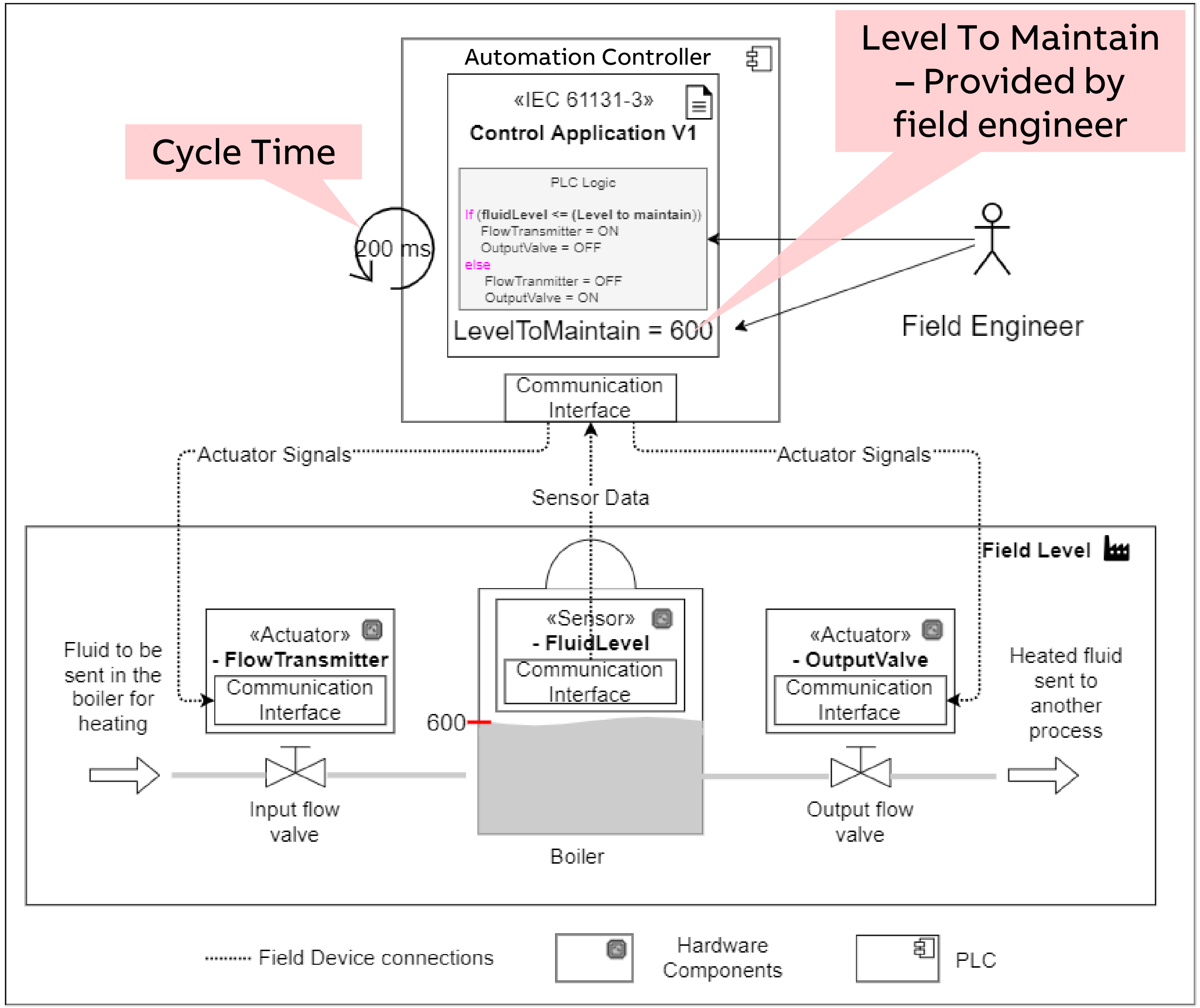


Industrial Control Application for Maintaining Fluid Level in Boiler

- Variables

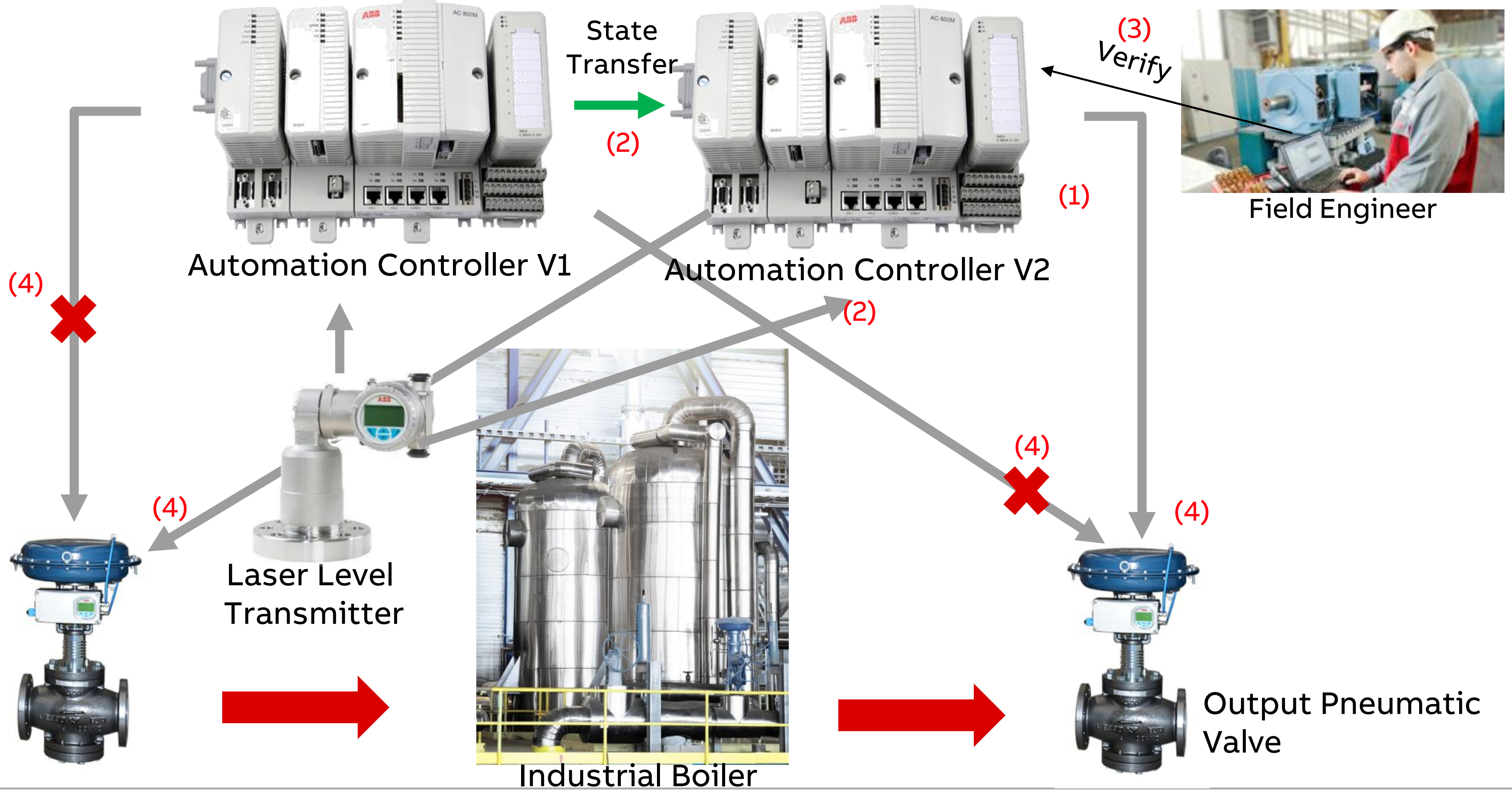
Variable	Type	
FluidLevel	INT	
LevelToMaintain	INT	(Retain)
FlowTransmitter	BOOL	
OutputValve	BOOL	

- Software Update Steps [1] [2]
  1. Initialization
  2. State Synchronization
  3. Verification
  4. Switch

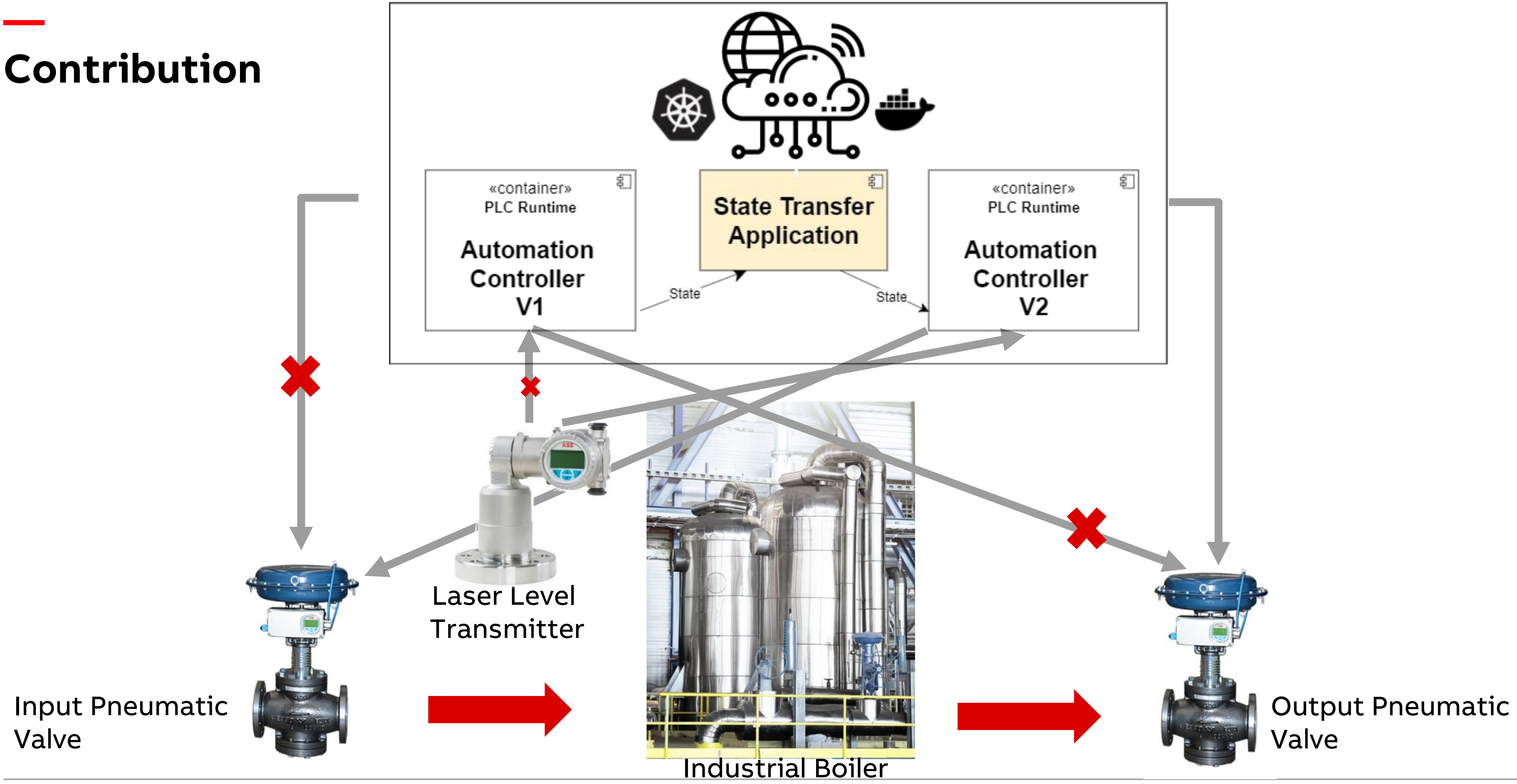


[1] :ABB (2015) System 800xA: Engineering and Production Environments [Online], ABB (3BSE045030-510 C). Available at https://library.e.abb.com/public/7de2b5acd288480695245919ceedc03b/3BSE045030510_C_en_System_800xA_Engineering_5.1_Engineering_and_Production_Environments.pdf (Accessed 25 July 2020).
[2] : Michael Wahler, Stefan Richter and Manuel Oriol (2009) 'Dynamic software updates for real-time systems', *Proceedings of the 2nd International Workshop on Hot Topics in Software Upgrades*, pp. 1–6.

Updating Industrial Control Application



Contribution

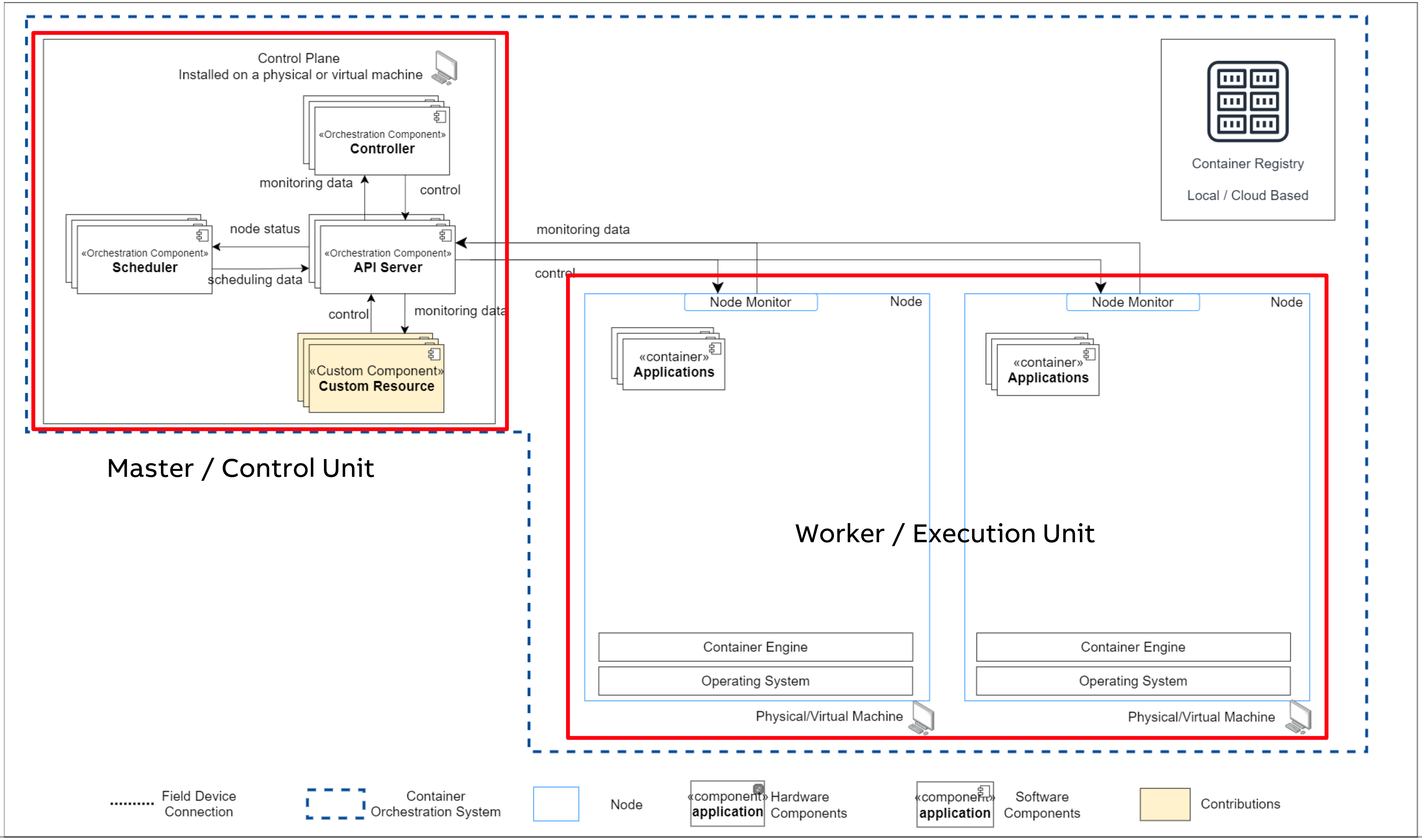


Research Questions

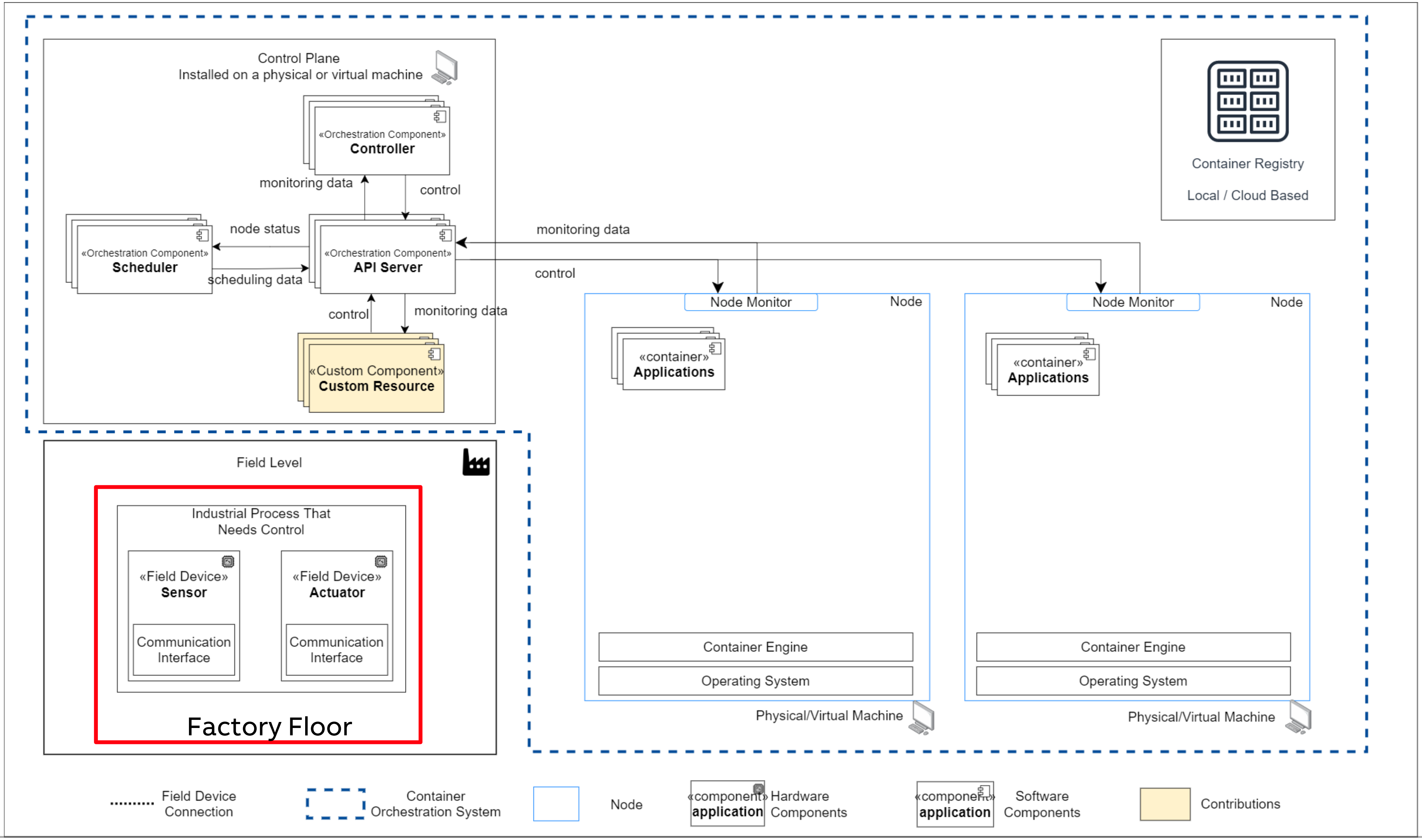
- What is the difference between updating a web application and an Industrial control application ?
- How will the internal state transfer work in cloud-native environment ?
- How can the existing features of container orchestration systems can be reused ?
- Which update strategies can be used ?
- How would manual and automatic updates work ?

2. Concept

Overview

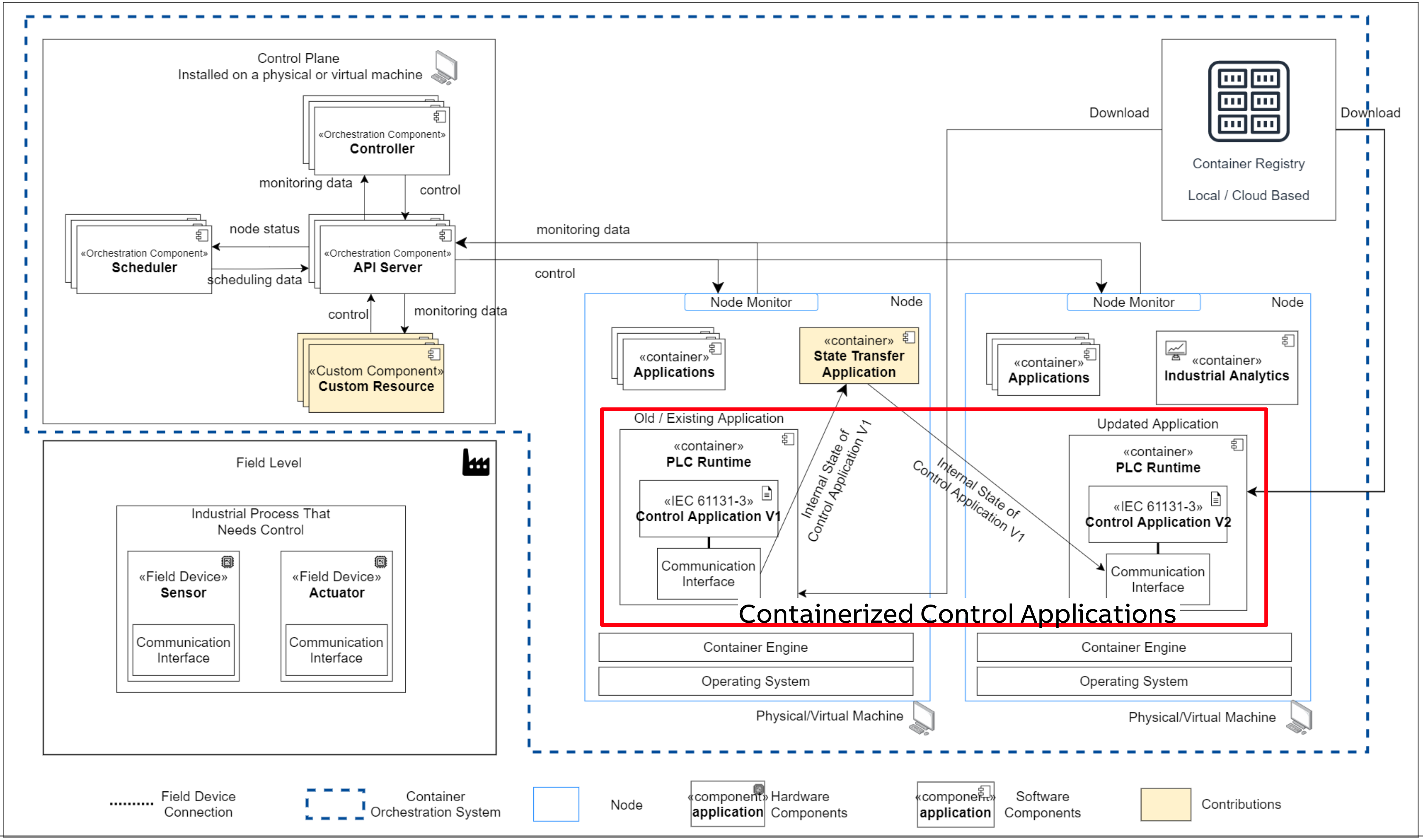

Control Plane
Installed on a physical or virtual machine
«Orchestration Component»
Controller
monitoring data
control
«Orchestration Component»
Scheduler
node status
scheduling data
«Orchestration Component»
API Server
control
monitoring data
«Custom Component»
Custom Resource
Master / Control Unit
Container Registry
Local / Cloud Based
monitoring data
control
Node Monitor
Node
«container»
Applications
Container Engine
Operating System
Physical/Virtual Machine
Node Monitor
Node
«container»
Applications
Container Engine
Operating System
Physical/Virtual Machine
Worker / Execution Unit
Field Device Connection
Container Orchestration System
Node
«component»
application
Hardware Components
«component»
application
Software Components
Contributions

Overview

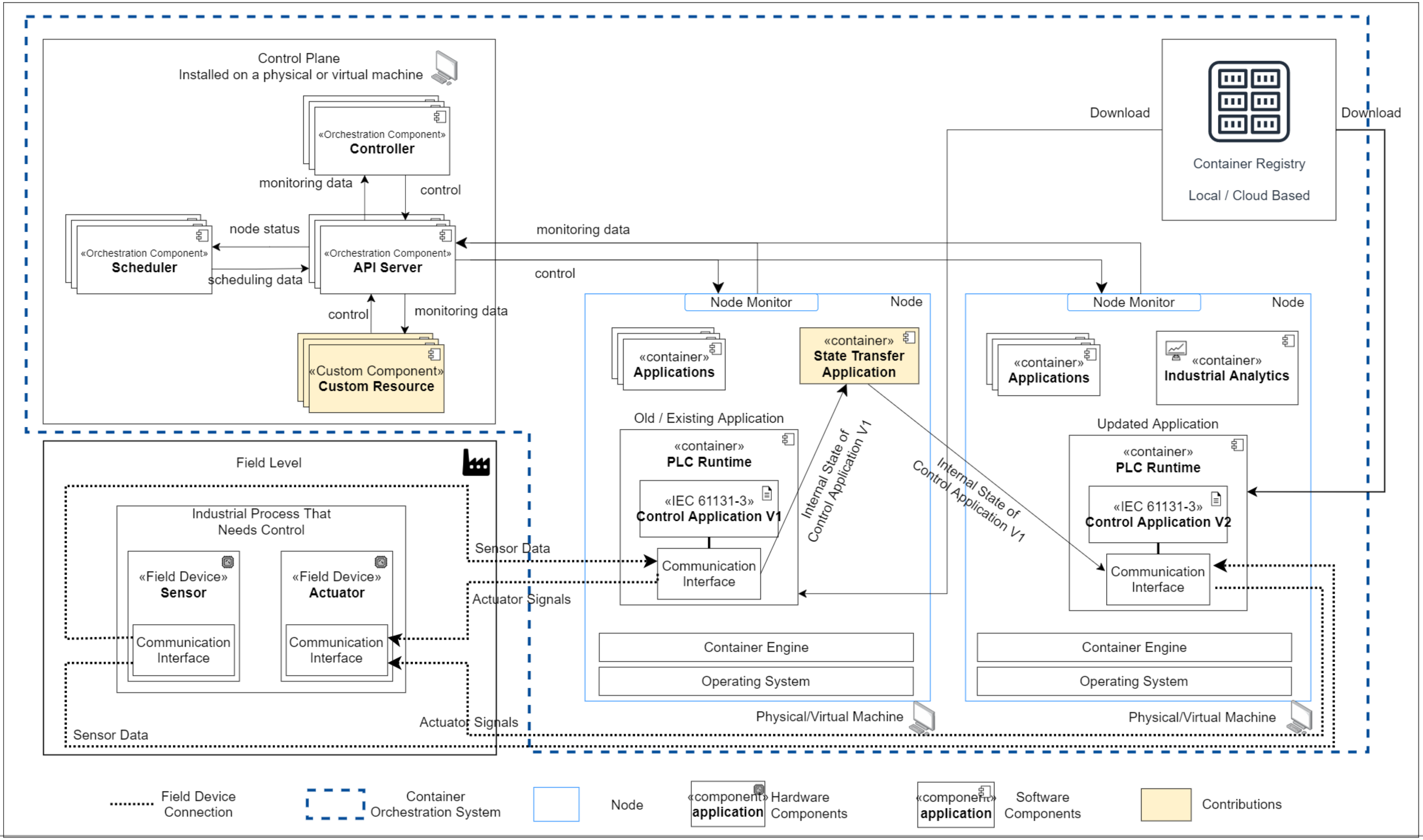


Control Plane
Installed on a physical or virtual machine
«Orchestration Component»
Controller
monitoring data
control
node status
«Orchestration Component»
Scheduler
scheduling data
«Orchestration Component»
API Server
control
monitoring data
«Custom Component»
Custom Resource
monitoring data
control
Container Registry
Local / Cloud Based
Node Monitor
Node
«container»
Applications
Container Engine
Operating System
Physical/Virtual Machine
Node Monitor
Node
«container»
Applications
Container Engine
Operating System
Physical/Virtual Machine
Field Level
Industrial Process That
Needs Control
«Field Device»
Sensor
Communication
Interface
«Field Device»
Actuator
Communication
Interface
Factory Floor
Field Device
Connection
Container
Orchestration System
Node
«component»
application
Hardware
Components
«component»
application
Software
Components
Contributions

Overview


Control Plane
Installed on a physical or virtual machine
«Orchestration Component»
Controller
monitoring data
control
node status
«Orchestration Component»
Scheduler
«Orchestration Component»
API Server
scheduling data
control
monitoring data
«Custom Component»
Custom Resource
Field Level
Industrial Process That
Needs Control
«Field Device»
Sensor
Communication Interface
«Field Device»
Actuator
Communication Interface
Download
Container Registry
Local / Cloud Based
Download
monitoring data
control
Node Monitor
Node
«container»
Applications
«container»
State Transfer Application
Old / Existing Application
«container»
PLC Runtime
«IEC 61131-3»
Control Application V1
Communication Interface
Internal State of Control Application V1
Internal State of Control Application V1
Node Monitor
Node
«container»
Applications
«container»
Industrial Analytics
Updated Application
«container»
PLC Runtime
«IEC 61131-3»
Control Application V2
Communication Interface
Containerized Control Applications
Container Engine
Operating System
Physical/Virtual Machine
Container Engine
Operating System
Physical/Virtual Machine
Field Device Connection
Container Orchestration System
Node
«component» application
Hardware Components
«component» application
Software Components
Contributions

Overview



Control Plane
Installed on a physical or virtual machine
«Orchestration Component»
Controller
monitoring data
control
node status
«Orchestration Component»
Scheduler
scheduling data
«Orchestration Component»
API Server
control
monitoring data
«Custom Component»
Custom Resource
Download
Download
Container Registry
Local / Cloud Based
monitoring data
control
Node Monitor
Node
«container»
Applications
«container»
State Transfer Application
Old / Existing Application
«container»
PLC Runtime
«IEC 61131-3»
Control Application V1
Communication Interface
Internal State of Control Application V1
Internal State of Control Application V1
Node Monitor
Node
«container»
Applications
«container»
Industrial Analytics
Updated Application
«container»
PLC Runtime
«IEC 61131-3»
Control Application V2
Communication Interface
Container Engine
Operating System
Physical/Virtual Machine
Container Engine
Operating System
Physical/Virtual Machine
Field Level
Industrial Process That Needs Control
«Field Device»
Sensor
Communication Interface
«Field Device»
Actuator
Communication Interface
Sensor Data
Actuator Signals
Sensor Data
Actuator Signals
Field Device Connection
Container Orchestration System
Node
«component»
application
Hardware Components
«component»
application
Software Components
Contributions

State Transfer Application and Slack Time

Slack Time

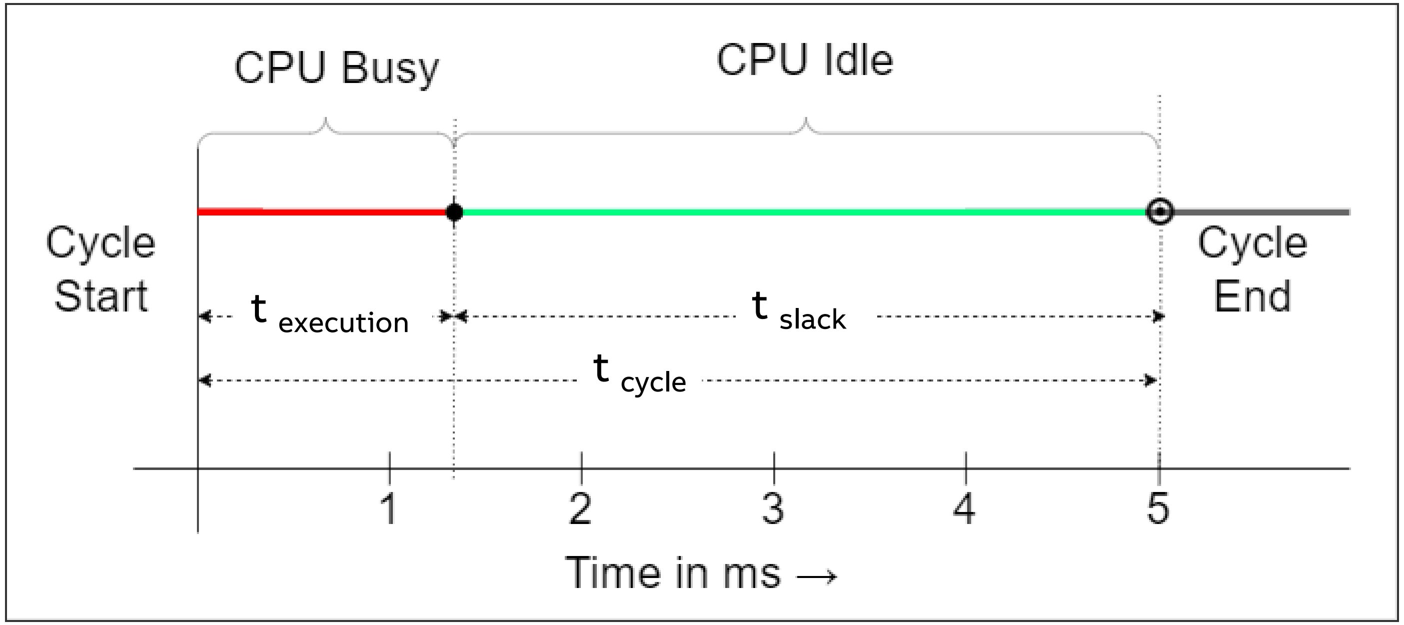
Slack Time is Time during which CPU is Idle

$t_{slack} = t_{cycle} - t_{execution}$

t_{slack} = Slack Time
t_{cycle} = Cycle Time of Control Application
$t_{execution}$ = Actual Execution time of one

Key Requirement [1], [2]:

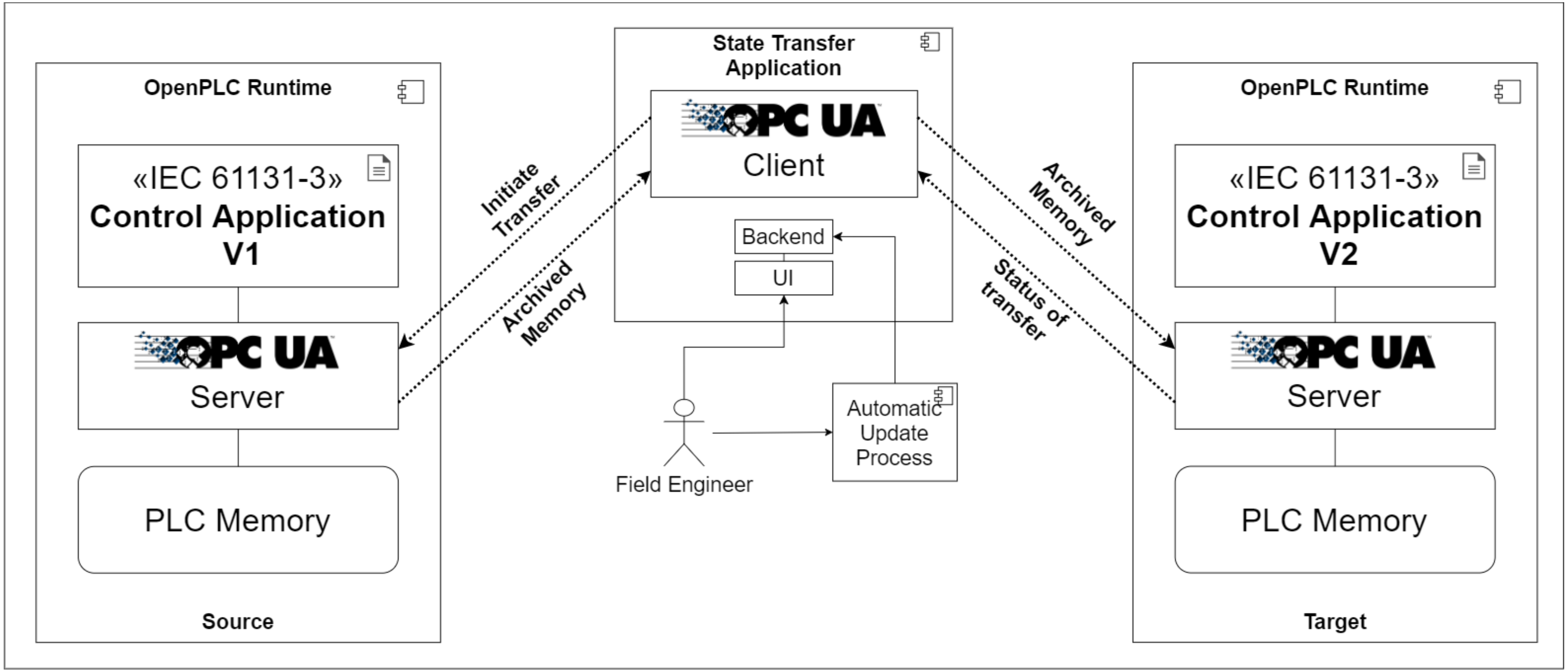
State Transfer Time < Slack Time



[1] : Michael Wahler; Stefan Richter; Manuel Oriol (2009): Dynamic software updates for real-time systems. In : Proceedings of the 2nd International Workshop on Hot Topics in Software Upgrades, pp. 1–6.
[2] : Michael Wahler; Thomas Gamer; Atul Kumar; Manuel Oriol (2015): FASA: A software architecture and runtime framework for flexible distributed automation systems. In *Journal of Systems Architecture* 61 (2), pp. 82–111. DOI: 10.1016/j.sysarc.2015.01.002.

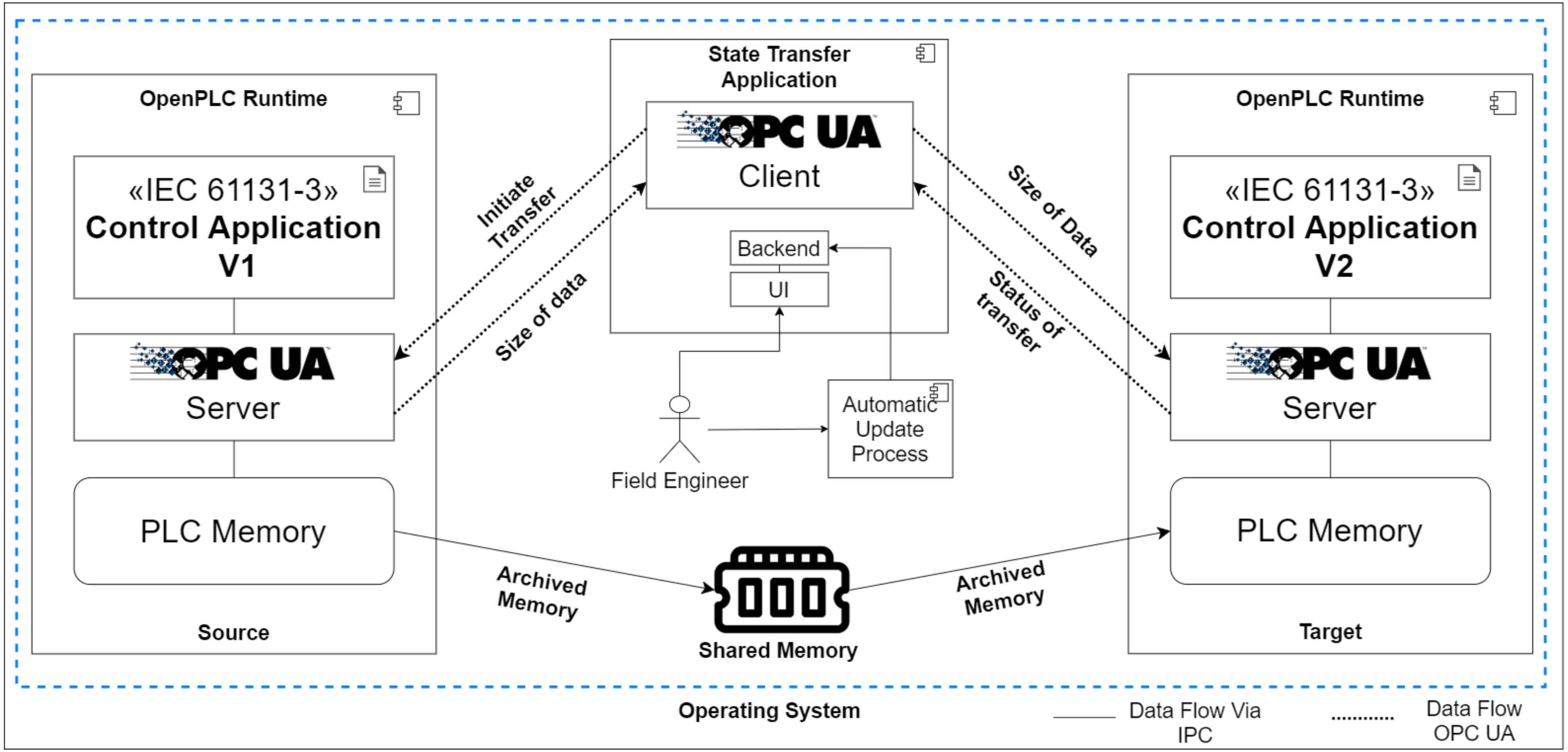
State Transfer via Network

State Transfer using OPC UA

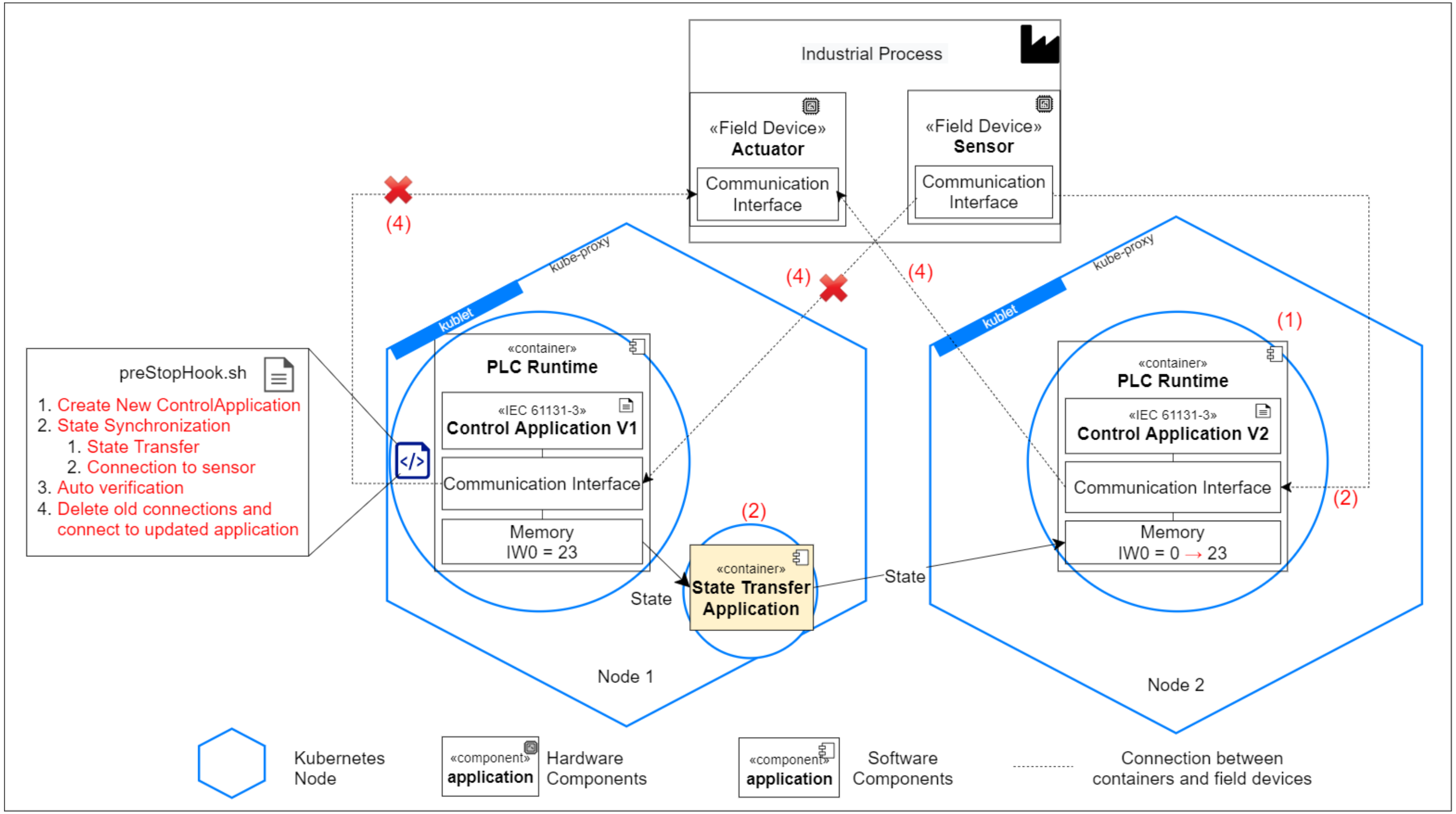


State Transfer via Shared Memory

State Transfer using Inter-Process Communication (IPC) via POSIX Shared Memory in Linux



Automatic Update Using Pre-Stop Container Lifecycle Hook



Pros

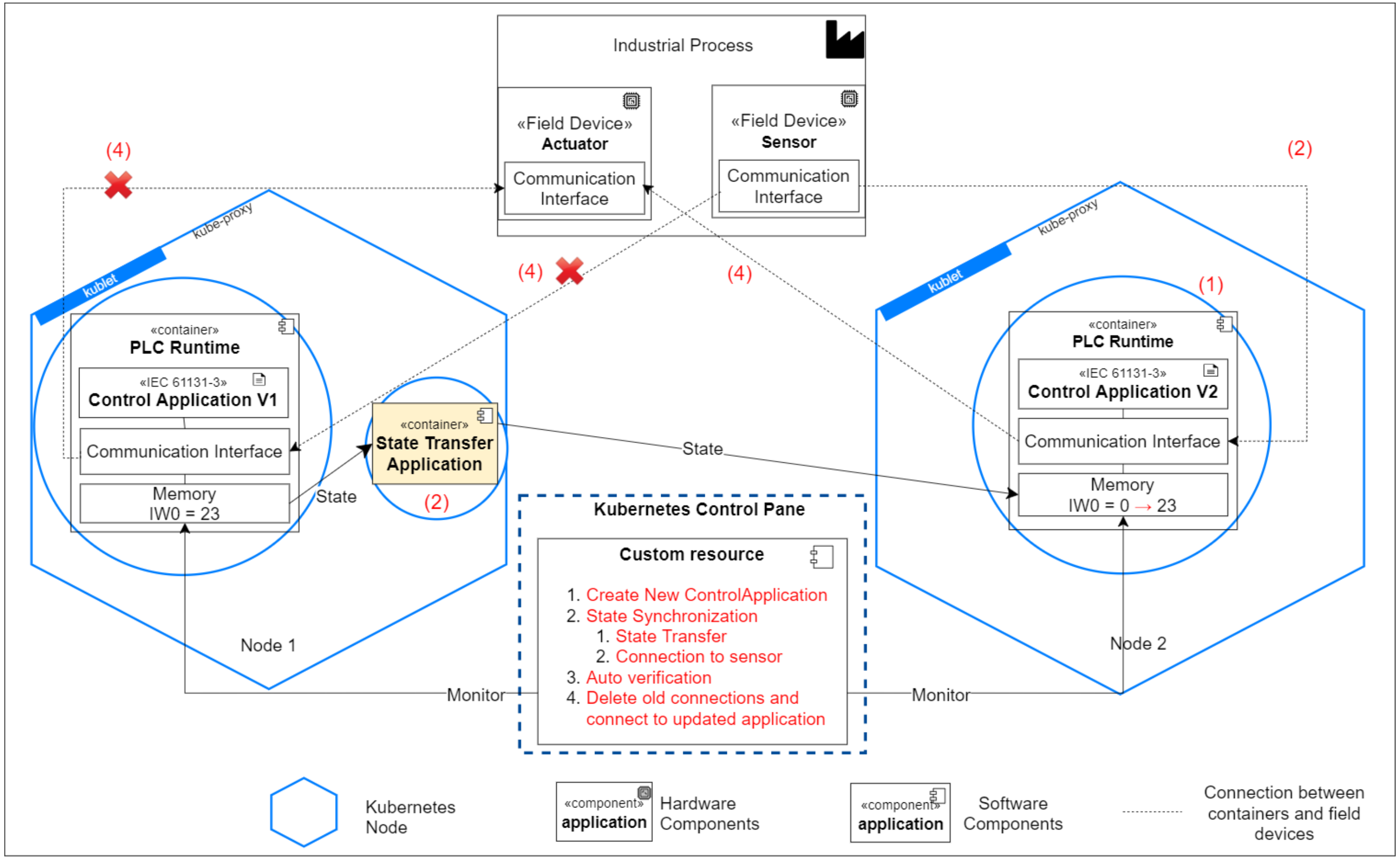
- Easy to implement with existing features
- Built in support by Kubernetes.

Cons

- Rollback support not available
- Less control to user as runs within a container

HOCHSCHULE HEIDELBERG SRH Intelligence in Learning

Automatic Update Using Custom Resources



Pros

- More control to the user
- Rollback can be implemented

Cons

- Complex implementation
- Needs expert in cloud native systems to develop custom resources

SRH HOCHSCHULE HEIDELBERG Intelligence in Learning ABB

Overview of Update Scenarios

Use Case	Example	State transfer required	Docker Image Change	Occurrence (Frequency)	Type of update (Manual / Automatic)
Updates in Control application	Update in **logic** or **functionality** of the **control application**	Yes	Yes	**Development – Frequent** (Weekly) **Production – Rare** (2-3 Years)	Manual OR Automatic depending on the type of logic change
	Minor Updates in underlying **operating system** of the **container** running Control Application	Yes	Yes	**Frequent** (2-3 months) [1]	Automatic
	Update in the **Communication Interface** (OPC UA Version) of the Control Application	Yes	Yes	**Frequent** (2-3 months) [2]	Automatic
	Change in the environment variables (IP address for field devices)	Yes	No	**Rare** (1 year to several years)	Automatic
Change in the field device.	**Replacement** of a **field device** due to malfunctioning	No	No	**Rare** (1 year to several years)	Manual
Update in Kubernetes nodes	Update in **Kubernetes Version** of the node	Yes	No	**Rare** (1 year) [3]	Manual (Requires automatic restart for control applications)
	Major Update in **host operating system** (Real Time Patching, OS Level Security Updates)	Yes	No	**Rare** (1-3 Years) [4]	Manual (Requires automatic restart for control applications)

[1] Debian (2020): Debian - Homepage. Debian. Available online at https://www.debian.org/, checked on 8/6/2020.
[2] open62541 (2020): Open62541 - Releases. Available online at https://github.com/open62541/open62541/releases, checked on 8/9/2020.
[3] https://github.com/kubernetes/community/blob/master/contributors/design-proposals/release/versioning.md
[4]: https://rt.wiki.kernel.org/index.php/Main_Page

SRH HOCHSCHULE HEIDELBERG Intelligence in Learning ABB

3. Results

Typical Control Applications

Examples :

- Example 1 : Liquified Natural Gas (LNG) Plant in Norway [1]
  - 650,000 Variables (approx. 5 MB)
  - More than 18 Control Units
  - 500ms Cycle Time
- Example 2 : Train Control Management System by Bombardier Transportation [2]
  - 4 to 84 Variables (approx. 0.1 kB)

Typical Requirements

- Variable Size = 10 – 650,000 Variables (0.1kb – 5MB) -> 25 – 30% Retained
- Cycle Time = 0.5 – 500 ms [3]

References
[1] Krause, Herbert (2007): Virtual commissioning of a large LNG plant with the DCS 800XA by ABB. In : 6th EUROSIM Congress on Modelling and Simulation, Ljubljana, Slovénie.
[2] Muslija, Adnan (2017): On the complexity measurement of industrial control software. Malardalen University, Vasteras, Sweden. Available online at https://www.diva-portal.org/smash/get/diva2:1113035/FULLTEXT01.pdf, checked on 10/7/2020.
[3] Gangakhedkar, Sandip; Cao, Hanwen; Ali, Ali Ramadan; Ganesan, Karthikeyan; Gharba, Mohamed; Eichinger, Josef (2018): Use cases, requirements and challenges of 5G communication for industrial automation. In : 2018 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, pp. 1–6

Testing Data

Tests Performed

- Data Range = 36 kb to 2.7 MB
- Variable Range = 23,552 to 1,725,000

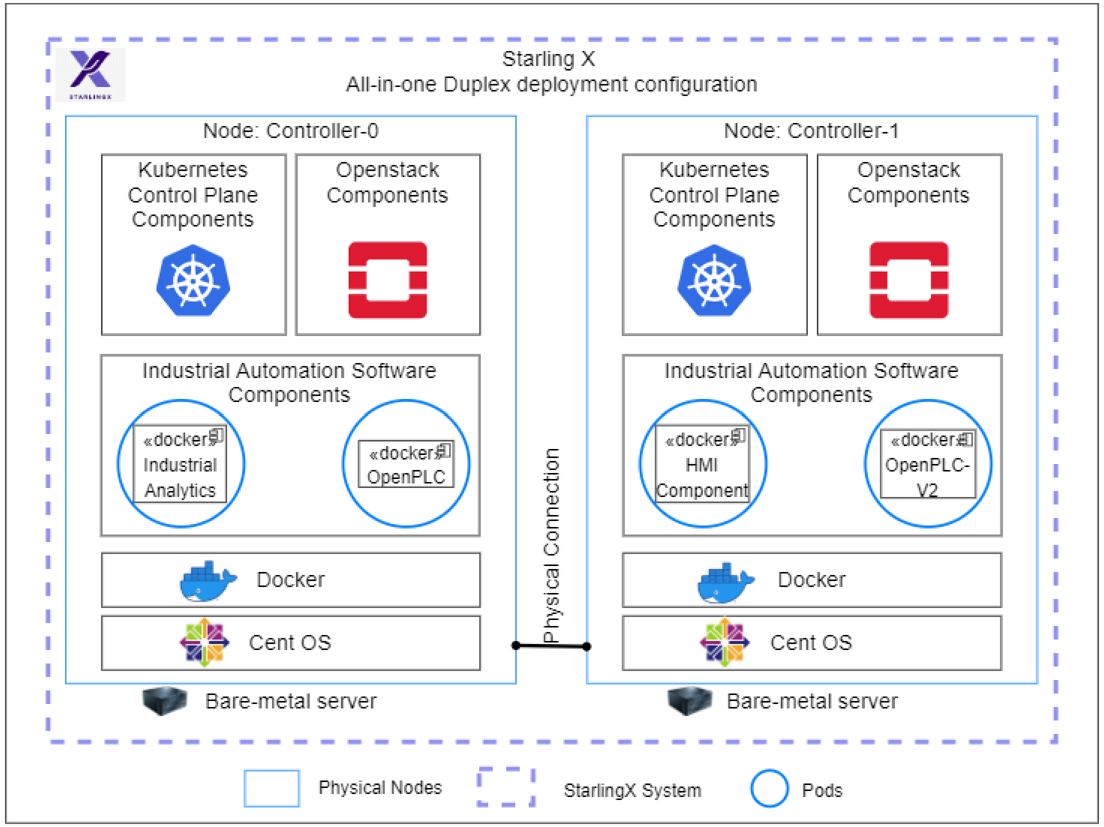
Data Size	Buffer Size	Input Integers	Output Integers	Input Booleans	Output Booleans	Input Bytes	Output Bytes	Memory Integers	Memory Doubles	Memory Long	Total Variables
36 kb	1024	1024	1024	1024*8	1024*8	1024	1024	1024	1024	1024	23,552
180 kb	5000	5000	5000	5000*8	5000*8	5000	5000	5000	5000	5000	115,000
360 kb	10000	10000	10000	10000*8	10000*8	10000	10000	10000	10000	10000	230,000
540 kb	15000	15000	15000	15000*8	15000*8	15000	15000	15000	15000	15000	345,000
720 kb	20000	20000	20000	20000*8	20000*8	20000	20000	20000	20000	20000	460,000
1.08 MB	30000	30000	30000	30000*8	30000*8	30000	30000	30000	30000	30000	690,000
1.44 MB	40000	40000	40000	40000*8	40000*8	40000	40000	40000	40000	40000	920,000
1.88 MB	50000	50000	50000	50000*8	50000*8	50000	50000	50000	50000	50000	1,150,000
2.7 MB	75000	75000	75000	75000*8	75000*8	75000	75000	75000	75000	75000	1,725,000

Test Infrastructure

- 2 Identical Machines With
  - Intel Xenon CPU E5-2640 v3, Frequency 2.60 GHz, 8 Cores
  - 64 GB RAM Each
  - CentOS 7.6
  - Starling X version 3.0
  - Kubernetes version 1.16.2

- Both Machines Provide Controller, Worker, Storage Functionality

- Programmable Logic Controller
  - OpenPLC – An Opensource software-based PLC



State Transfer Time Results

Use Cases :

- opcua-same-node :

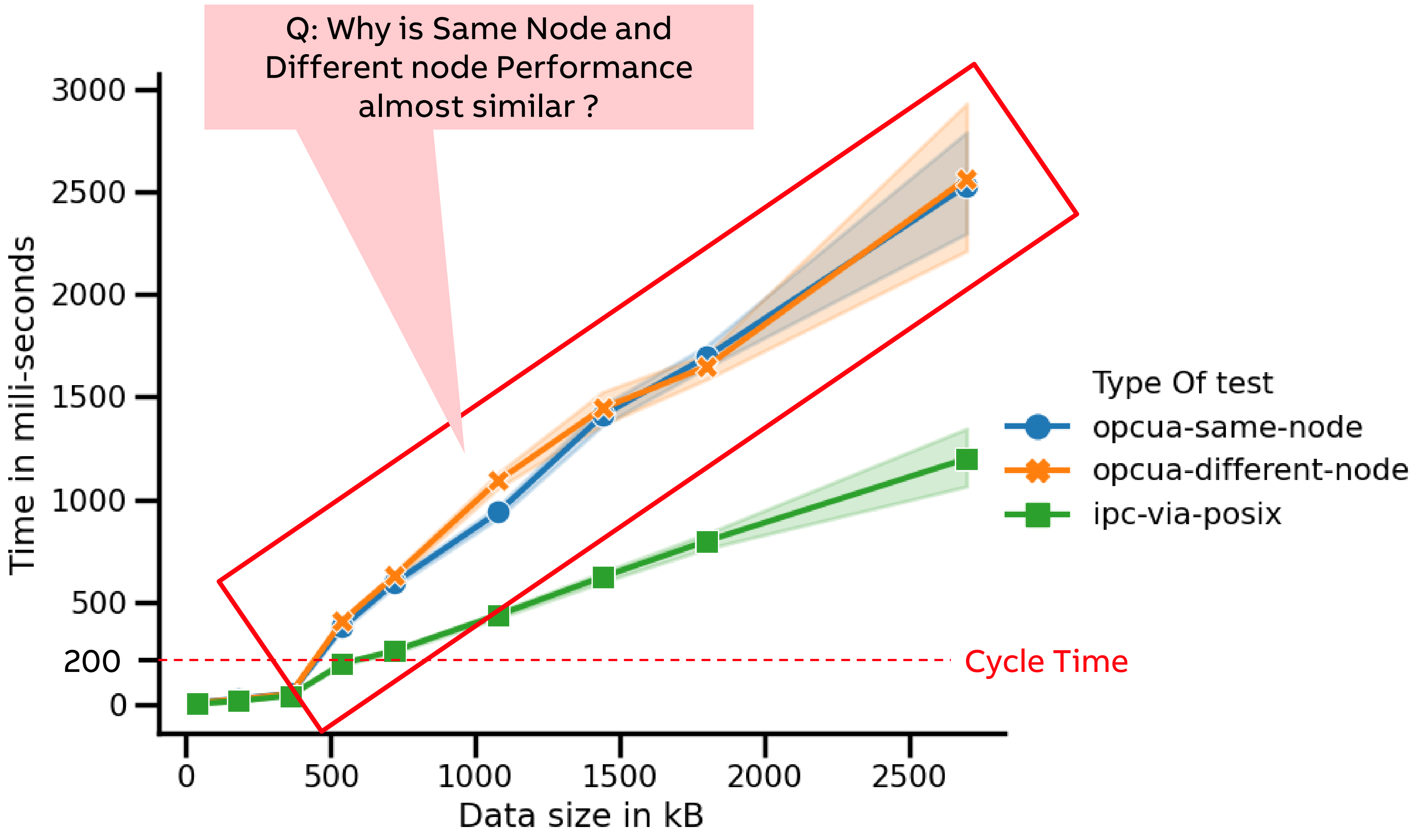
State Transfer via **Network** using OPC UA with source and target control application in the **same physical node**.

- opcua-different-node :

State Transfer via **Network** using OPC UA with source and target control application in the **different physical node**.

- ipc-via-posix :

State transfer using Inter-Process Communication via POSIX **shared memory** in linux



Analysis of results

Table : Average State Transfer Time and Slack Time for Different State Sizes

Data Size	Variables	Average PLC execution time ms	Cycle Time in ms	Slack Time in ms	Average State Transfer Time in ms		
					OPC UA Same Node	OPC UA Different Node	IPC via POSIX Shared Memory
36 kb	23 k	0.0833	200	~ 200	9.2616	7.7406	3.4984
180 kb	115 k	0.0593	200	~ 200	27.3745	25.2997	16.9567
360 kb	230 k	0.1245	200	199.88	52.7567	50.6416	43.0406
540 kb	345 k	-	200	-	377.1343	407.5989	199.8418
720 kb	460 k	-	200	-	593.3057	626.2894	261.4155
1.08 MB	690 k	-	200	-	938.2413	1091.7047	436.8433
1.44 MB	920 k	-	200	-	1409.1815	1444.2984	621.8367
1.8 MB	1150 k	-	200	-	1694.9999	1647.6269	796.8021
2.7 MB	1725 k	-	200	-	2530.9898	2565.5177	1198.8593

Analysis of results

State Transfer Time < Slack Time

Table : Average State Transfer Time and Slack Time for Different State Sizes

Data Size	Variables	Average PLC execution time ms	Cycle Time in ms	Slack Time in ms	Average State Transfer Time in ms		
					OPC UA Same Node	OPC UA Different Node	IPC via POSIX Shared Memory
36 kb	23 k	0.0833	200	~ 200	9.2616	7.7406	3.4984
180 kb	115 k	0.0593	200	~ 200	27.3745	25.2997	16.9567
360 kb	230 k	0.1245	200	199.88	52.7567	50.6416	43.0406
540 kb	345 k	-	200	-	377.1343	407.5989	199.8418
720 kb	460 k	-	200	-	593.3057	626.2894	261.4155
1.08 MB	690 k	-	200	-	938.2413	1091.7047	436.8433
1.44 MB	920 k	-	200	-	1409.1815	1444.2984	621.8367
1.8 MB	1150 k	-	200	-	1694.9999	1647.6269	796.8021
2.7 MB	1725 k	-	200	-	2530.9898	2565.5177	1198.8593

State Transfer via Network vs State Transfer using Shared Memory

State Transfer via Network (State Transfer using OPC UA)	State Transfer using Shared Memory (IPC via POSIX Shared Memory)
Transfer across multiple nodes is possible	Transfer across different nodes is not possible
Network Overhead causes slower transfer speeds	No network overhead so faster state transfer
No requirement of source and target containers in same pod	IPC via POSIX transfer only works Single pod with two containers, as kubernetes as POSIX memory is shared only between containers running in the same pod.
Simple and straightforward setup	Complex setup

4. Conclusion

Research Questions

- How will the internal state transfer work in cloud-native environment ?
  - State Transfer Application
- How can the existing features of container orchestration systems can be reused ?
  - Container Lifecycle Hooks
- How would manual and automatic updates work ?
  - State Transfer Application and UI, Container Lifecycle Hooks and Custom Resource Definition

Conclusion and Future Work

This thesis presents

- A study of existing update strategies used in container orchestration systems
- A novel technique for performing disruption-free software updates to industrial control applications running in container orchestration systems using a state transfer application
- Demonstrates that the solution is feasible and can work in the long run.

During this thesis, some future challenges were identified which can be studied in future works :

- Using Custom Resources in container orchestration systems for performing Automatic updates
- State Transfer across multiple cycles
- Design of a User Interface for State Transfer Application for Better User Experience

ABB