Configuration of Dependable Edge Computing Platforms for Virtualized Critical Control Applications

Paul Pop

2023-10-05

DTU Compute
Department of Applied Mathematics and Computer Science

Outline and acknowledgements

- Motivation
- Industry 4.0: IT/OT convergence
- Dependable Edge Computing
  - Edge vs. Fog Computing
  - Virtualization of control
- Configuration problems
  - Mapping and scheduling of tasks
  - Routing and scheduling of messages
  - Fault-tolerance, quality of control, extensibility and security
- Configuration optimization solutions
- Evaluation and use case

This presentation is based on the PhD theses of the following DTU PhD students:

- Dr. Mohammadreza Barzegaran
- Dr. Niklas Reusch

The work has been done in collaboration with:

- TTTech Computertechnik AG
- Danfoss Power Electronics A/S

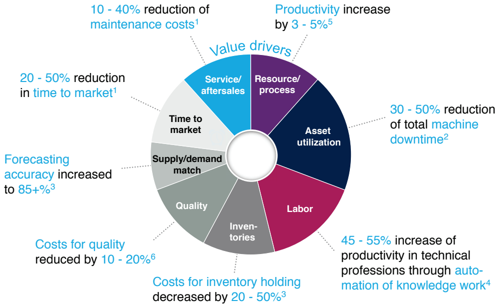
The work has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 764785: FORA—Fog Computing for Robotics and Industrial Automation

Motivation

Industry 4.0: *Digitalization of the manufacturing and industrial sectors, with embedded sensors in virtually all product components and manufacturing equipment, ubiquitous cyber-physical systems, and analysis of all relevant data.*

Benefits (manufacturing)

Indicative quantification of value drivers



Source: McKinsey

Digitalization of all areas

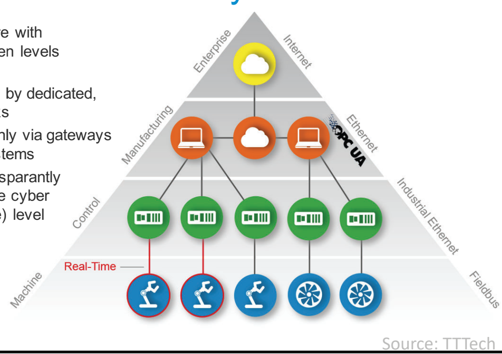
Source: Overview The Internet Of Things (IoT) System Security, Applications, Architecture And Business Models

The automation pyramid and OT

Operations technology (OT): The hardware and software dedicated to detecting or causing changes in physical processes through direct monitoring and/or control of physical devices such as valves, pumps, etc.

The Automation Pyramid: Soon to be Ancient History?

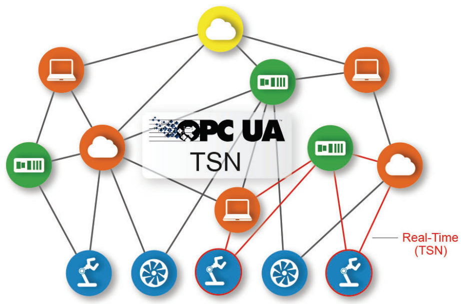
- Rigid infrastructure with separation between levels of functionality
- Levels connected by dedicated, specialist networks
- Data exchange only via gateways or proprietary systems
- Difficulties to transparantly access data at the cyber pyramid (machine) level



Source: TTTech

From the automation pyramid to distributed architectures

- Cyber-Physical Systems are integrated into manufacturing, logistics, industrial processes.
- Production complexity moves to the Cloud: benefits cost, energy, sharing of resources, flexibility, adaptability.
- Open standards will win: IEEE 802.1 TSN and OPC UA over TSN

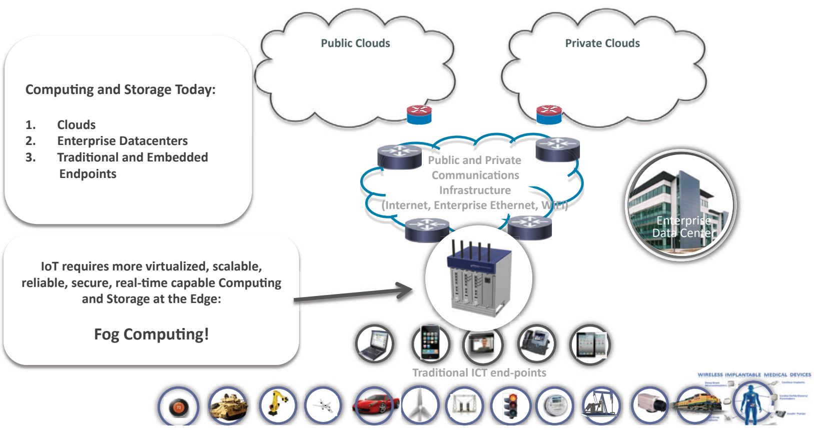


Source: TTTech

What is Cloud that comes close to the ground (edge of the network)?

Fog Computing System-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things.

Similarities with: (Dependable Real-Time) Edge Computing

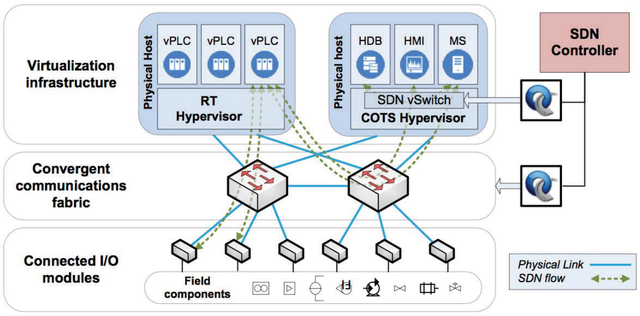


Source: Flavio Bonomi

Vision: OT becomes virtualized (Control-as-a-Service)

- **Edge Computing:** An architecture where resources of an edge server are placed at the very edge of the Internet, near Cyber-Physical Systems (CPSs), mobile devices, sensors, and Industrial Internet of Things (IIoT) endpoints.
- Benefits: Reduced latency, bandwidth efficiency, enhanced privacy & security.

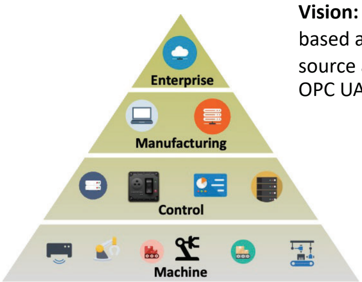
- **Dependable Edge Computing:** Ensuring reliable, fault-tolerant, and secure data processing at the edge, especially for critical applications.
- Benefits: Enhanced robustness, reduced risks, trustworthy operations.



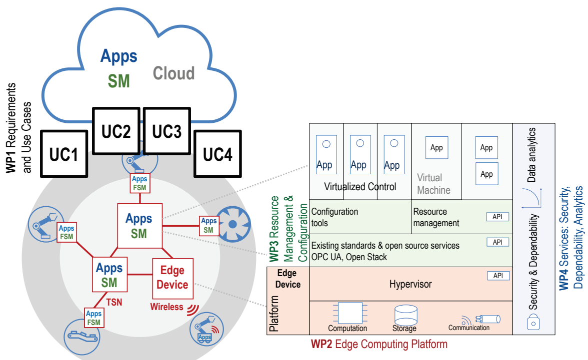
FORA: Fog Computing for Robotics and Industrial Automation

- **The FORA European Training Network**
  - Marie Skłodowska-Curie Action
  - 15 PhD students, 5 universities, 4 countries

- **DTU's objectives in FORA: virtualization of control**
  - Develop methods and tools for the configuration of an Edge Computing Platform for critical control applications



Vision: open Edge Computing-based architecture built on open source and open standards (TSN, OPC UA and 5G).

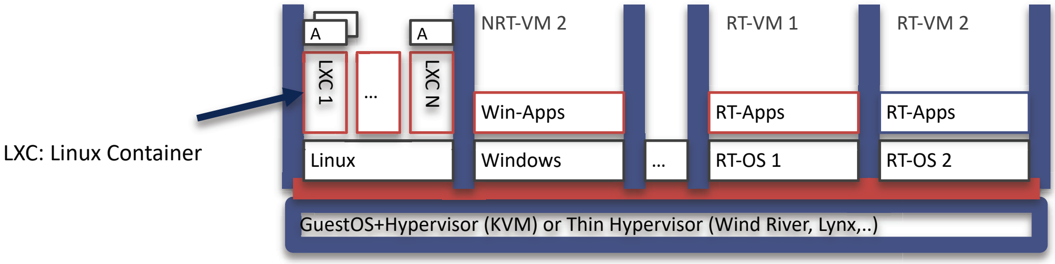


Current closed architectures using the automation pyramid (Purdue model)

Industry 4.0 architecture based on Dependable Edge Computing and Deterministic Networking (TSN)

Edge nodes: Virtualization and Security

Virtualization A combination of physical separation (multicore), hard, RT-NRT Virtual Board/ Machine based virtualization and more lightweight Linux Container based virtualization



Security Decentralized model, including distributed authentication, trusted booting, secure software management, SDN based connectivity control, OpenVPN, NFV Security…

Network convergence: IEEE Time-Sensitive Networking

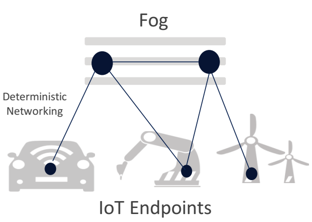
Principles **Integration**

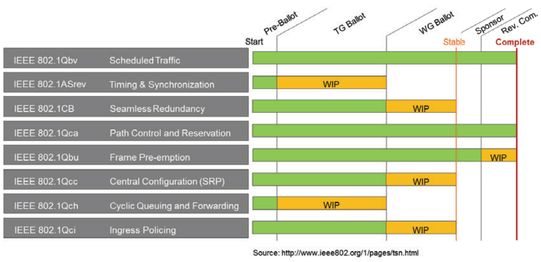
Multiple traffic classes share the network, supporting applications with mixed-criticality requirements

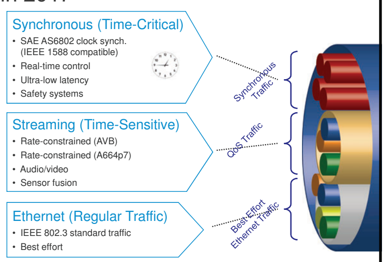
Separation: Virtual links separate different criticalities

Standardization as IEEE TSN

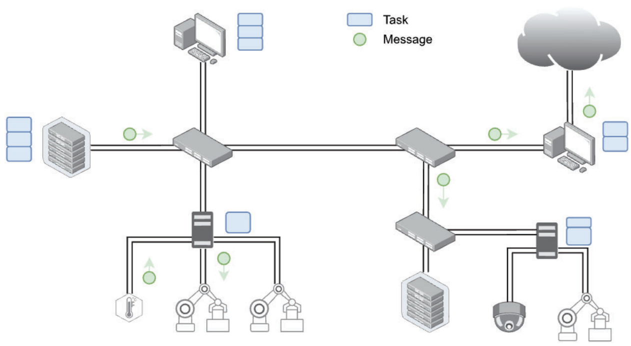
A number of IEEE 802.1 standards due for release early in 2017







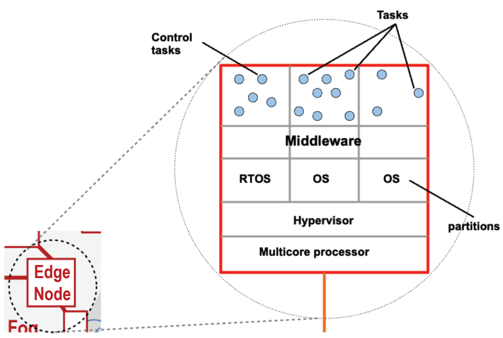
Motivation: Example Architecture



→ Engineering dependable edge-based systems for critical control is challenging:
- Architecture design, provisioning and traffic shaping mechanisms
- Placement and scheduling of tasks, routing and scheduling of messages
- Fault-tolerance, redundancy, security mechanisms
- Resource management, guaranteeing timeliness, safety, security

Edge Computing Platform Architecture

- **The Edge Computing Platform (ECP) runs mixed-criticality applications, including control applications**
  - An FCP is composed of several interconnected Edge Nodes (ENs), from powerful multicore ENs to low-end ENs
  - The control applications are virtualized as tasks running on the Edge Nodes (ENs) of the ECP
  - Partitioning is used to isolate applications of different criticalities
    - Each partition can have its own operating system (OS)
    - The partitions running the control applications use a real-time operating system (RTOS)

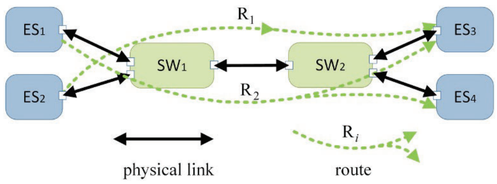


Example architecture of an Edge Node

Time-Sensitive Networking

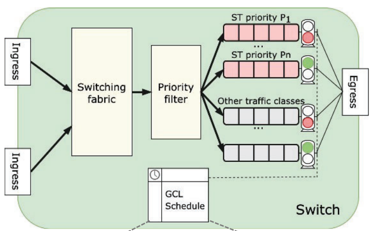
- Set of amendments to IEEE 802.1:
  - Extends Ethernet with **real-time** and **safety-critical capabilities**
  - Applied in many application areas
- Consists of many sub-standards; we consider:
  - **802.1ASrev:** time synchronization
  - **802.1Qbv:** time-aware traffic shaper
- A TSN network is composed of:
  - End Systems (ES) with input and output ports
  - Switches (SW) with input and output ports
  - Links between ES and SW
- The traffic is described by streams:
  - Sender, receivers, data size, period
  - Packed into Ethernet frames
  - Traffic type and priorities (**TT**, AVB, BE)

Standard	Description
802.1AS-Rev	Timing and Synchronization for Time-Sensitive Applications
802.1CB	Frame Replication and Elimination for Reliability
802.1Qbv	Enhancements for Scheduled Traffic, Time Aware Shaper
802.1Qbu	Frame Preemption
802.1Qci	Per-Stream Filtering and Policing



TSN 802.1Qbv Switch Architecture

- Ingress & Egress **Ports**
- 8 **priority queues**
- Queues can be assigned to different traffic classes
- **Switching Fabric** (and **Priority Filter**) map frames to priority queues
- **Gates** control flow of traffic: Opened/closed based on **Gate-Control List (GCL)**
- TSN also offers other shapers

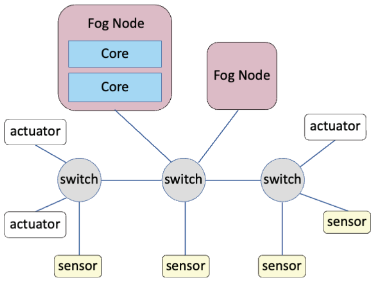


	Gate Open Vector
[0,10]	10000000
...	...
...	...
...	...
...	...
[90,100]	00000001

Architecture model

Architecture model

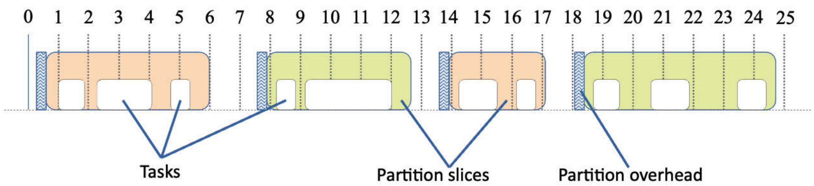
- Set of Fog nodes connected via TSN; each Fog Node has
  - Multiple cores
  - A hypervisor and a time-triggered scheduler
    - Isolation of mixed-criticality applications



Example architecture model

Partition tables and scheduling

- Partitions are statically scheduled using partition tables (e.g., as in PikeOS)
- Time-triggered scheduling is used to run the tasks, and communication is ignored (we'll revisit this in Paper B)

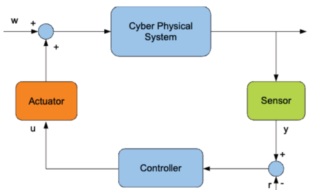


Example schedule table

Control applications and their performance

A **Feedback control system (FCS)** or control application operates and commands a dynamical system (robots and industrial machines) using a control algorithm.

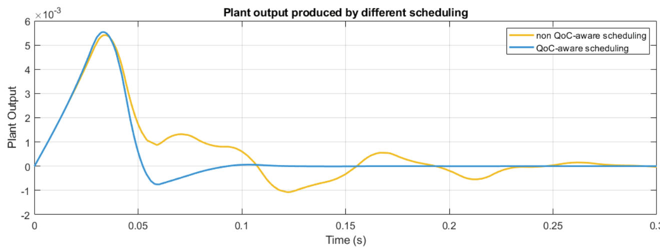
- Can be implemented as a three-task application: sampling, control law, and actuation tasks



A simple feedback control system

Control performance (QoC)

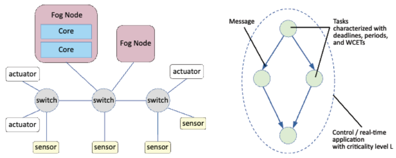
- Captures the trade-off between the accuracy and the rapidity of the controller
- We use a quadratic cost function (J) proposed in the literature
- QoC is calculated using the JitterTime tool, which simulates control applications



Scheduling effect on the control output

A. Cervin, P. Pazzaglia, M. Barzegaran, and R. Mahfouzi, "*Using JitterTime to analyze transient performance in adaptive and reconfigurable control systems*," In Proceeding of IEEE International Conference on Emerging Technologies and Factory Automation, pp. 1025-1032, 2019.

ECP configuration for QoC: tasks

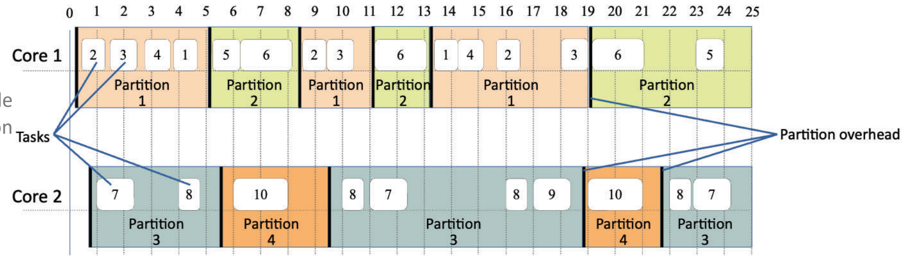


Given: Application and architecture models

Determine: An ECP configuration:

- partitions
- mapping of tasks to the cores
- assignment of tasks to partitions
- the period of control applications
- partition tables
- task schedule tables

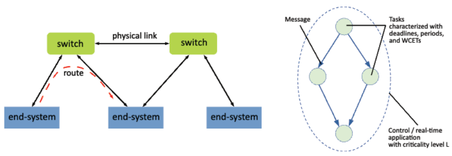
Example configuration



Such that:

- The QoC of applications is maximized and “balanced” across applications
- The mixed-criticality applications are isolated within partitions, and the deadlines are satisfied

ECP configuration for QoC: messages

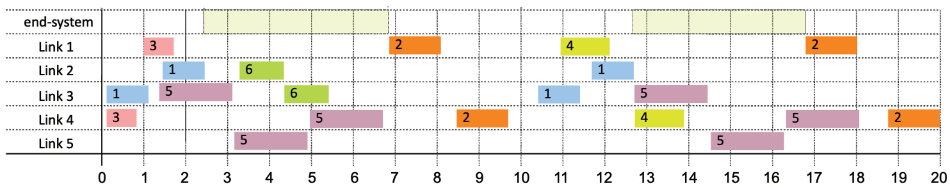


Given: Application and architecture models

Determine: An ECP configuration

- Including the GCLs for messages

Example configuration

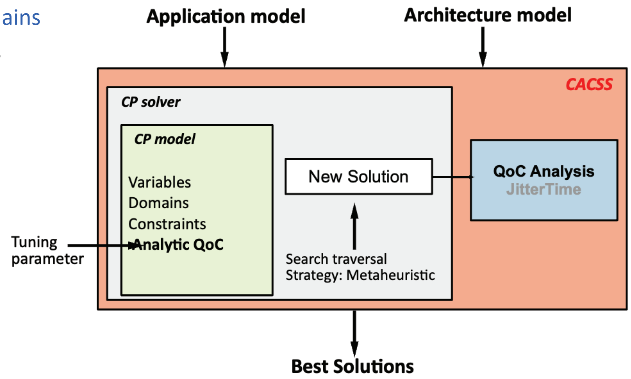


Such that:

- The QoC of control applications is maximized
  - The QoC is captured using an analytic model
- The deadlines for mixed-criticality applications are satisfied

Control-Aware Configuration Strategy

- CP model consist of a set of variables and their domains
  - The variable domains are related by constraints



- Constraint Programming (CP) is a declarative programming paradigm
- CP visits solutions that satisfy the constraints and evaluates them using an objective function
- The visited solutions are evaluated using the analytic QoC model
- For each improving solution JitterTime calculates the accurate QoC value

Evaluation results

Fog Computing Platform Configuration (FCPC) has 3 variants:

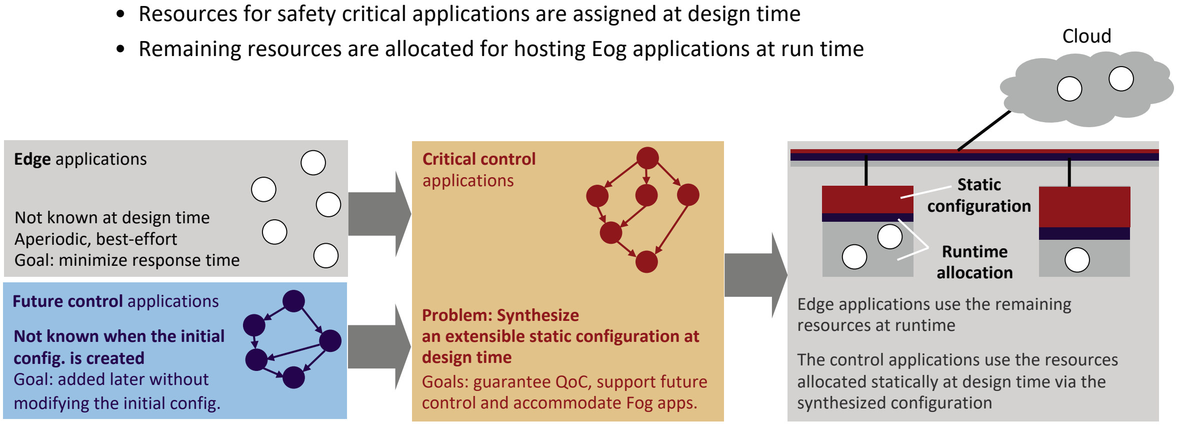
1. FCPC/M—ignores mapping optimization
2. FCPC/Q—ignores the QoC optimization
3. FCPC/P—ignores the control applications' period optimization

Test cases	No. of Cores	Total No. of Control Applications	Total No. of Tasks	Total No. of Tasks for Criticality level of [0-4]	Ω of FCPC	Ω for FCPC/M	Ω for FCPC/Q	Ω for FCPC/P
1	2	2	12	{1,2,2,1,6}	0,19	48%	Not Feasible	60%
2	2	3	23	{3,2,5,4,9}	0,34	5%	Not Feasible	Not Feasible
3	2	2	17	{2,2,4,3,6}	0,21	5%	Not Feasible	33%
4	2	2	23	{2,1,7,6,7}	0,29	13%	Not Feasible	13%
5	3	3	32	{1,4,8,8,11}	0,21	39%	Not Feasible	55%
6	3	3	31	{2,3,9,7,10}	0,22	85%	Not Feasible	50%
7	3	4	33	{4,4,8,7,12}	0,26	15%	Not Feasible	10%
8	4	4	44	{4,6,11,9,14}	0,20	29%	Not Feasible	72%
9	5	6	54	{7,5,14,10,18}	0,21	21%	Not Feasible	Not Feasible
10	6	7	63	{6,7,16,12,22}	0,24	21%	Not Feasible	46%

Ω is the value of the cost function; a small value means better and well-balanced QoC for control applications

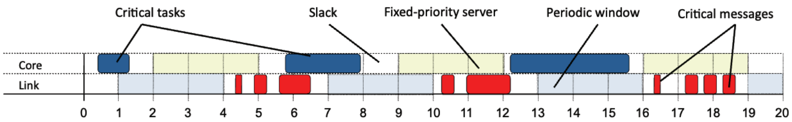
Extensibility in Edge Computing

- **The ECP runs mixed-criticality applications; safety-critical applications require static pre-release configuration**
  - The vision is Industry 4.0 is to host Edge applications, which are not considered at design time
  - A costly re-certification is required when the configuration is changed
  - An extensible configuration is used in the ECP to realize Industrial IoT
    - Resources for safety critical applications are assigned at design time
    - Remaining resources are allocated for hosting Eog applications at run time

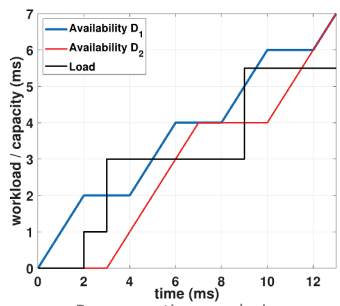


Extensible ECP configuration

- **An extensible ECP configuration is synthesized at design time and considers changes in runtime.**
  - The changes can be future critical applications, Edge applications or both; handled by appropriate technique put together in a hierarchical scheduling model
  - The extensible schedule accommodates a larger number of future control applications and provides a shorter response time for Edge applications
- **The extensible configuration uses the uniform distribution of the periodic slack in schedules to host the changes.**
  - The slack in the schedules is distributed uniformly at design time
  - The slack is used to allocate resources for dimensioned servers to handle applications



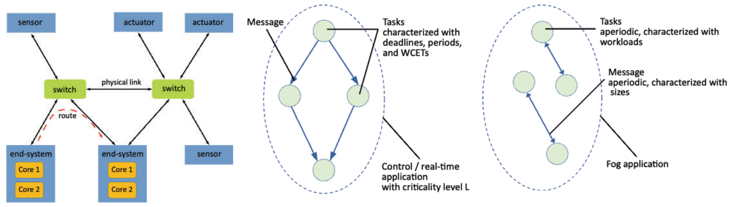
Extensible configuration



Response time analysis

ECP configuration for extensibility

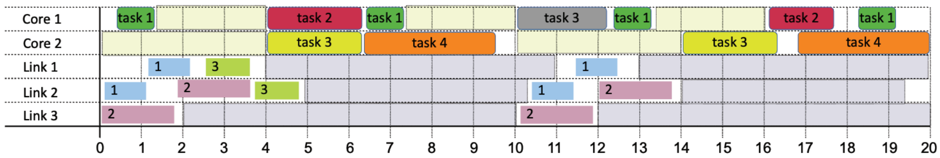
Given: Application and architecture models



Determine: An ECP configuration:

- mapping of tasks to the cores
- routes for critical control applications
- Gate Control Lists (GCLs)
- static task schedule tables
- dimensioning of deferrable servers
- dimensioning of port windows

Example configuration



Such that:

- The QoC of applications is maximized and "balanced" across applications
- The deadlines are satisfied
- The extensibility of the ECP configuration is maximized

Extensible Configuration Optimization Strategy (ECOS)

Architecture model

Application model

ECOS models the problem as a CP model
- The CP model consists of a set of variables
  - Start time of frames
  - End time of frames
  - Start time of jobs
  - End time of jobs
  - Mapping of tasks to the cores of FNs

ECOS defines a CP formulation for the problem
- The CP formulation consists of a set of constraints
  - Link overlapping
  - Routing
  - Isolation of frames
  - Frame deadlines
  - Core utilization
  - Task overlapping
  - Task deadlines
  - Precedence in applications

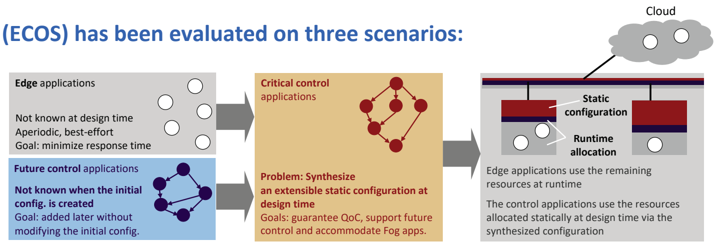
ECOSS generates a set of optimized solutions

- ECOS improves the speed of the search with a metaheuristic search traversal strategy
- The strategy uses heuristic methods for choosing variables and assigning values to the variables

Evaluation results

Extensible Configuration Optimization Strategy (ECOS) has been evaluated on three scenarios:

1. Supporting future control applications
2. Hosting Edge applications
3. Extending with upgrades



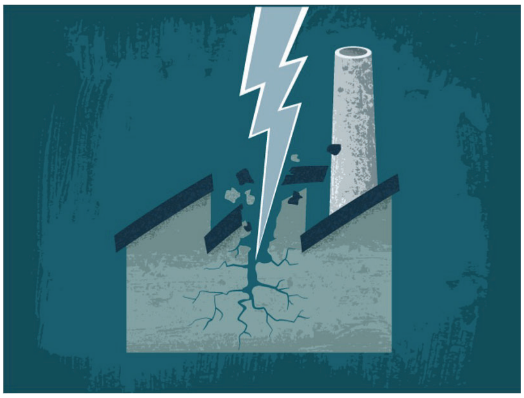
TC #	Total no. of tasks / flows in FCCAs[1]	Mean util. of FCCAs[1]	Percentage of Supported FCCAs[1]		RT^2 of Fog application 1 Tasks: 16 Flows: 15		RT^2 of Fog application 2 Tasks: 21 Flows: 20		RT^2 of Fog application 3 Tasks: 35 Flows: 38	
			ECOS	ECOS/E	ECOS	ECOS/E	ECOS	ECOS/E	ECOS	ECOS/E
1	36/58	57%	100%	78%	1.33	3.97	2.82	5.66	4.73	7.43
2	37/65	55%	100%	89%	1.42	1.91	2.75	4.92	6.74	9.56
3	30/20	50%	100%	96%	1.26	3.36	2.94	4.88	5.16	12.29
4	29/32	45%	100%	81%	2.17	3.27	3.64	5.65	5.18	7.34
5	33/40	44%	100%	96%	1.56	4.14	3.68	5.81	4.36	9.44
6	44/45	40%	100%	90%	1.47	2.96	3.14	4.17	4.96	5.82
7	37/35	39%	100%	83%	1.19	3.95	3.88	3.96	2.96	4.76
8	33/34	37%	100%	90%	2.18	2.93	3.14	5.84	2.84	9.64
9	25/28	31%	100%	98%	1.65	3.77	4.43	5.93	2.35	4.74
10	18/21	27%	100%	82%	2.37	3.79	4.82	5.97	2.85	6.68
Average			100%	87%	1.6	3.4	3.5	5.2	4.2	7.7

[1] Future Critical Control Application
[2] Response time in ms.

ECOS/E ignores extensibility

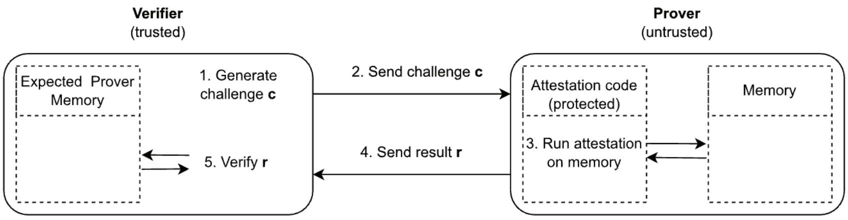
Example cybersecurity attack: Stuxnet

- Probably targeted at the nuclear program of Iran
- Used multiple zero-day vulnerabilities
- Manipulates PLCs in SCADA systems, commonly used in safety-critical systems
- Highly sophisticated & targeted:
  - Only attacks industrial drives from two vendors
  - Only attacks drives which spin at a certain frequency (gas centrifuges)
  - Slightly modifies frequency over period of months to increase wear but remain stealthy



Remote Attestation: Idea

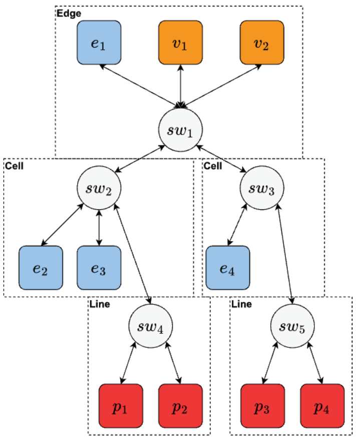
- Given:
  - Set of **trusted** end-systems called **verifiers**
  - Set of **untrusted** end-systems called **provers**
- Goal:
  - Validate the integrity of software on provers
  - Detect changes in software, e.g., because of malware
- Problem:
  - Attestation **takes long time** (multiple seconds)
  - **Can't be interrupted:** Infeasible in real-time systems
- Solution: SMART + SMARM architecture
  - Memory is split into equal-sized blocks
  - Only a certain block is attested each time
  - The prover cannot predict which block → malware cannot hide



Remote Attestation Process

Architecture Model

- Industrial-inspired Edge-Computing platform
  - Edge servers (data analytics, machine learning, etc.)
  - Cells (control)
  - Production lines (sensors and actuators)
- Edge devices are most powerful and best protected:
  Verifiers are placed in the edge devices
- Sensors and actuators on production lines are least protected:
  Provers are placed in the IoT endpoints

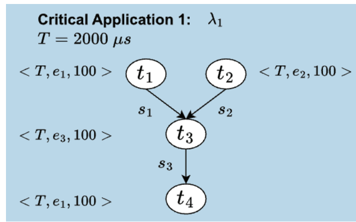


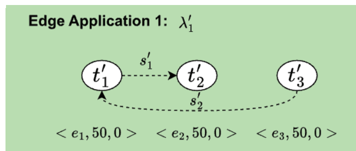
Application Model

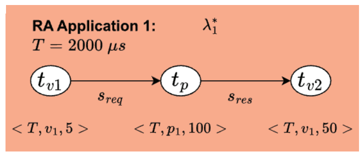
- **Critical control applications**
  - Hard real-time, periodic
  - DAG with tasks and streams
  - Unknown task placement & stream routing

- **Edge applications**
  - Not real-time, but response time is important
  - aperiodic (arrival time unknown)
  - No data dependencies, but
    async communication via streams

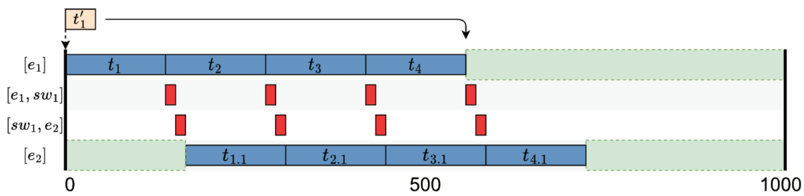
- **Remote attestation applications**
  - Consists of 3 tasks and 2 streams
  - Should happen as regular as possible



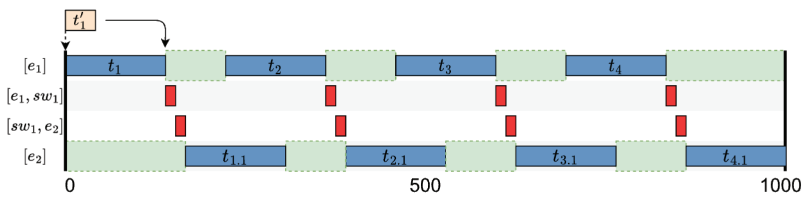




Example: Edge Application Optimization

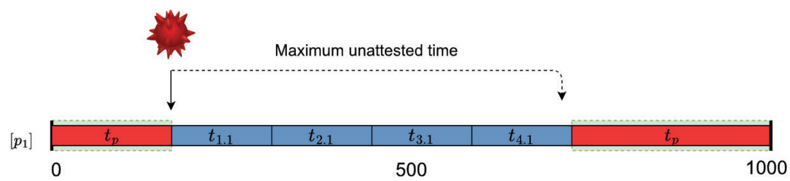


No optimization → long worst-case response time

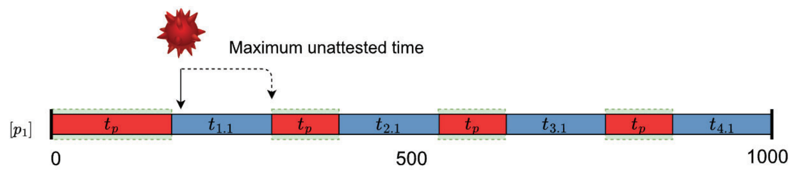


Optimization → shorter worst-case response time

Example: RA Application Optimization



No optimization → long worst-case detection time



Optimization → shorter worst-case detection time

ECP configuration for security

Given:

- Edge Computing TSN architecture, designated verifiers & provers
- Critical real-time applications
- A set of dynamic edge applications

Determine:

- Task mapping
- Static cyclic task schedules
- Stream routing & schedules (GCLs)
- RA applications and deferrable servers

Such that:

- Deadlines of critical real-time applications are met
- Response time for edge applications is minimized
- Security for RA applications is maximized

Evaluation

- Python, CP-SAT solver from Google OR-tools
- Medium-sized industrial inspired testcase
- **NOEXT**: scheduler without optimization for extensibility/security
- **EXT:** scheduler with optimization

➢ **EXT** significantly reduces avg. & worst-case response time for random edge applications
➢ **EXT** significantly reduces the maximum unattested time for all provers
➢ The benefits come at a slight increase in the latencies of critical applications (still schedulable)

	NOEXT	EXT
Maximum unattested time	425	150
Worst-case edge-app response time	350	233
Average-case edge-app response time	66,421	45,117
Total critical app latency	4,668	5,141

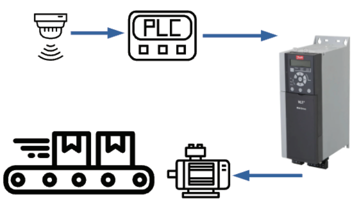
Comparison of NOEXT vs. EXT

	NOEXT	EXT
E1	121.75	79.75 (-34.5%)
E2	180.83	107.83 (-40.37%)
E3	215	185.17 (-13.87%)
E4	166.25	148.33 (-10.78%)
E5	161.67	134.67 (-16.7%)
E6	109.83	109 (-0.76%)
E7	101.5	65.33 (-35.64%)

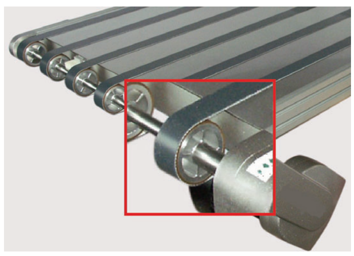
Response time for randomly-appearing edge applications (E1-E7)

Edge-based electric drives

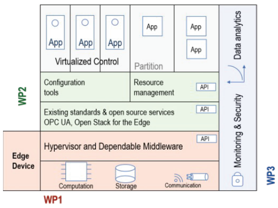
- **Electric drives are widely used in Industry**
  - They produce data that carries vital information: they can be used as the data source
- **Electric drives are re-engineered as ENs in an ECP**
  - Edge-based drives perform data analytics; avoids sending massive data with vital information
  - Electric drives are developed using the FORA FCP reference architecture
  - The architecture is modelled using Architecture Analysis & Design Language (AADL)

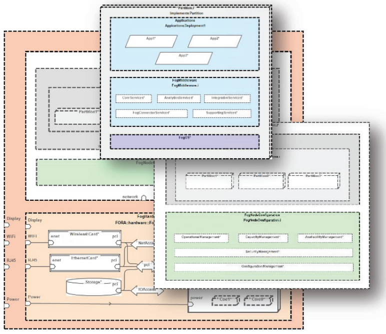


An electric drive in a conveyor belt setting

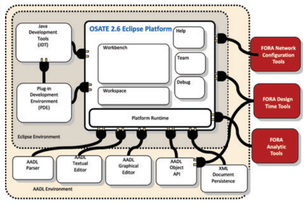

Conveyor belt

FORA AADL reference architecture





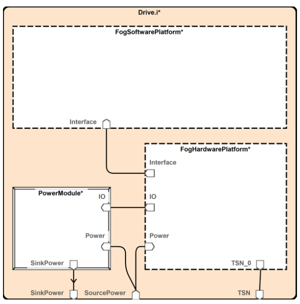
FORA AADL Models



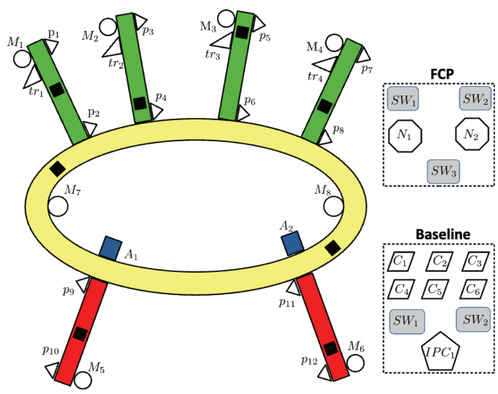
Use case architecture and model

Edge-based drives are modelled using Architecture Analysis and Design Language (AADL)

- AADL is a standard language for modelling systems using a component-oriented approach
- Edge-based drives are designed to deliver the drive punctualities and realize the vision of Industry 4.0



AADL model of the edge-based drive



UC schematics

A self baggage drop system

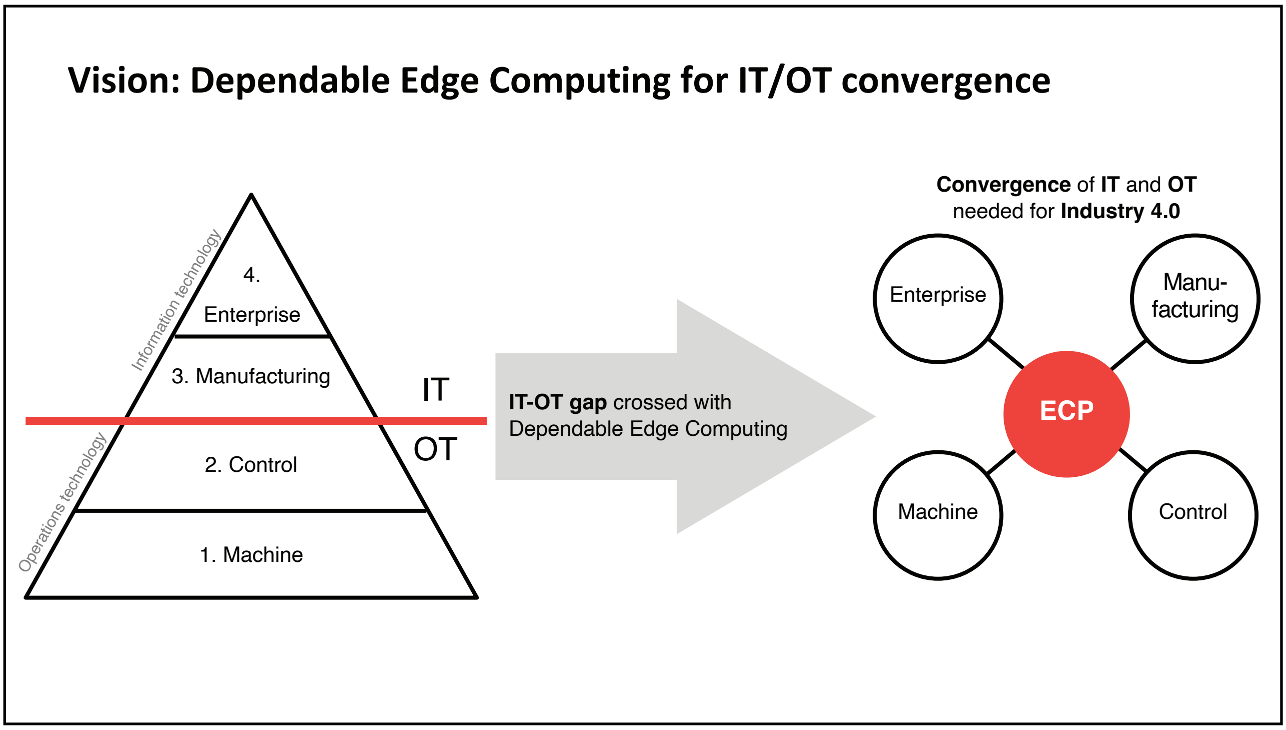
Research output and overall evaluation

Research contribution	KPIs						
	Increased safety	Increased security	Reduce latency of virtualized critical control	Reduction in installation, configuration and software management costs	Reduction in hardware spending	Increased access to machine data edge analytics	Shorter time-to-market for new industrial applications
Fog Node, Hypervisors and Middleware							
Clock synchronization for virtualization [36]	✓	●	✓	✗	●	✗	✗
Time-triggered hypervisor [17]	✓	✓	✓	●	●	✗	✗
Container reconfiguration technique [21]	✗	✗	✗	✓	●	✗	✓
Deterministic Communication							
Time-triggered networking [16], [37]	✓	●	●	✓	✗	✗	●
Wireless networking [38]	✗	●	✓	●	✗	✗	●
TSN configuration optimization [24]	●	✗	✓	●	●	✗	●
Resource Management, Orchestration and Configuration							
Orchestration method [19]	✗	✗	✗	●	●	●	✓
Routing configuration optimization [39]	✗	✗	✗	✓	✓	●	✓
Control virtualization method [40]	✓	✗	✓	●	●	✗	✗
Extensible configuration optimization [29]	✓	✗	✓	●	✓	●	✓
Application migration method [27]	✗	✗	●	✗	●	✗	✓
Scheduling algorithm for elastic applications [41]	●	✗	✓	✗	●	✗	●
Services and Applications							
Fault recovery mechanism [32]	✓	✗	✗	✗	✓	✗	✗
Fault-tolerant architecture [42]	✓	✗	●	●	✓	✗	✗
Authentication method [31]	●	✓	✗	●	✗	✗	✗
Intrusion detection method [18]	●	✓	✗	●	✗	✗	✗
Fault detection mechanism in TSN [33]	✓	✗	●	●	●	✗	✗
Safety services [43]	●	✗	●	●	●	✗	✗
Decomposed deep training solution [35]	✗	✗	✗	✗	✗	✓	●
Data analytics solution [34]	✗	✗	✗	✗	✗	✓	●

The public prototypes are at: http://www.fora-etn.eu/people/ and https://github.com/nreusch/TSNConf

Summary

- We proposed several approaches to the design time ECP configuration optimization for mixed-criticality applications
  - The configuration guarantees the performance and timeliness of control applications
  - The configuration provides maximum Quality-of-Service for dynamic Edge applications
  - The configuration consists of:
    - Decisions on the partitions that provide temporal and spatial isolation among mixed-criticality applications
    - Mapping the tasks to the cores of multicore Edge nodes
    - Routing of streams on TSN
    - Synthesizing the task schedule tables and GCLs
    - Remote Attestation using SMARM
- We proposed approaches to handle migration and best-effort scheduling of dynamic Fog applications at runtime
- We have developed several algorithms that use heuristics, metaheuristics and Constraint Programming to solve these combinatorial optimization problems
- We have proposed analytical models for QoC and extensibility that can be integrated to optimization strategies
- We have evaluated the algorithms on several test cases


Vision: Dependable Edge Computing for IT/OT convergence
Information technology
Operations technology
4.
Enterprise
3. Manufacturing
2. Control
1. Machine
IT
OT
IT-OT gap crossed with
Dependable Edge Computing
Convergence of IT and OT
needed for Industry 4.0
Enterprise
Manu-
facturing
ECP
Machine
Control