Configuration of Dependable Edge Computing Platforms for Virtualized Critical Control Applications

Paul Pop
2023-10-05

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 Skłodowska-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

<table>
<thead>
<tr>
<th>Value Driver</th>
<th>Percent Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintenance costs</td>
<td>10 - 40% reduction</td>
</tr>
<tr>
<td>Time to market</td>
<td>20 - 50% reduction</td>
</tr>
<tr>
<td>Forecasting accuracy</td>
<td>Increased by 85%</td>
</tr>
<tr>
<td>Costs for quality</td>
<td>Reduced by 10 - 20%</td>
</tr>
<tr>
<td>Costs for inventory holding</td>
<td>Decreased by 20 - 50%</td>
</tr>
<tr>
<td>Productivity increase</td>
<td>45 - 55% increase</td>
</tr>
<tr>
<td>Total machine downtime</td>
<td>30 - 50% reduction</td>
</tr>
<tr>
<td>Productivity increase</td>
<td>3 - 5% increase</td>
</tr>
</tbody>
</table>

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: 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 transparently access data at the cyber pyramid (machine) level

Source: TTech
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

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
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
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

- Synchronous (Time-Critical)
  - Robust deterministic behavior
  - Real-time control
  - Ultra-low latency
  - Safety-critical

- Streaming (Time-Sensitive)
  - Rate-sustained (RTP)
  - Rate-sustained (AAL5/T)
  - Audio/Video
  - Sensor Fusion

- Ethernet (Regular Traffic)
  - 802.3 standard
  - Best effort
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)
**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)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.1AS Rev</td>
<td>Timing and Synchronization for Time-Sensitive Applications</td>
</tr>
<tr>
<td>802.1CB</td>
<td>Frame Replication and Elimination for Reliability</td>
</tr>
<tr>
<td>802.1Qbv</td>
<td>Enhancements for Scheduled Traffic, Time Aware Shaper</td>
</tr>
<tr>
<td>802.1Gbu</td>
<td>Frame Preemption</td>
</tr>
<tr>
<td>802.1Qs</td>
<td>Per-Stream Filtering and Policing</td>
</tr>
</tbody>
</table>

![Diagram of 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

---

<table>
<thead>
<tr>
<th>Gate Open Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,10)</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>[90,100]</td>
</tr>
<tr>
<td>00000001</td>
</tr>
</tbody>
</table>
**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

**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)

**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

**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

---

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

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

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

<table>
<thead>
<tr>
<th>Test cases</th>
<th>No. of Cores</th>
<th>Total No. of Control Applications</th>
<th>Total No. of Tasks</th>
<th>Total No. of Criticality level of [0-4]</th>
<th>( \Omega ) of FCPC</th>
<th>( \Omega ) for FCPC/M</th>
<th>( \Omega ) for FCPC/Q</th>
<th>( \Omega ) for FCPC/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>(1, 2, 2, 1, 6)</td>
<td>0.19</td>
<td>48%</td>
<td>Not Feasible</td>
<td>60%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>23</td>
<td>(3, 2, 5, 4, 9)</td>
<td>0.34</td>
<td>5%</td>
<td>Not Feasible</td>
<td>5%</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>17</td>
<td>(2, 2, 4, 3, 6)</td>
<td>0.21</td>
<td>5%</td>
<td>Not Feasible</td>
<td>33%</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>23</td>
<td>(2, 1, 7, 6, 7)</td>
<td>0.29</td>
<td>13%</td>
<td>Not Feasible</td>
<td>13%</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>32</td>
<td>(1, 4, 8, 8, 11)</td>
<td>0.21</td>
<td>39%</td>
<td>Not Feasible</td>
<td>55%</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>31</td>
<td>(2, 3, 9, 7, 10)</td>
<td>0.22</td>
<td>85%</td>
<td>Not Feasible</td>
<td>50%</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>33</td>
<td>(4, 4, 8, 7, 12)</td>
<td>0.26</td>
<td>15%</td>
<td>Not Feasible</td>
<td>10%</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
<td>44</td>
<td>(4, 6, 11, 9, 14)</td>
<td>0.20</td>
<td>29%</td>
<td>Not Feasible</td>
<td>72%</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>6</td>
<td>54</td>
<td>(7, 5, 14, 10, 18)</td>
<td>0.21</td>
<td>21%</td>
<td>Not Feasible</td>
<td>46%</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>7</td>
<td>63</td>
<td>(6, 7, 16, 12, 22)</td>
<td>0.24</td>
<td>21%</td>
<td>Not Feasible</td>
<td>Not Feasible</td>
</tr>
</tbody>
</table>

\( \Omega \) 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 Fog 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
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

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)

- 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

- 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

**ECOSS** generates a set of optimized solutions
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

<table>
<thead>
<tr>
<th>TC #</th>
<th>Total no. of tasks / flows in FCCAs¹</th>
<th>Mean util of FCCAs²</th>
<th>Percentage of Supported FCCAs³</th>
<th>RT of Fog application 1 Tasks: 16 Flows: 15</th>
<th>RT of Fog application 2 Tasks: 21 Flows: 20</th>
<th>RT of Fog application 3 Tasks: 35 Flows: 36</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36/38</td>
<td>57%</td>
<td>100% 78%</td>
<td>1.33 3.97</td>
<td>2.82 5.66</td>
<td>4.73 7.43</td>
</tr>
<tr>
<td>2</td>
<td>37/65</td>
<td>55%</td>
<td>100% 69%</td>
<td>1.42 1.41</td>
<td>2.75 4.92</td>
<td>6.24 9.56</td>
</tr>
<tr>
<td>3</td>
<td>50/20</td>
<td>50%</td>
<td>100% 96%</td>
<td>1.26 3.16</td>
<td>2.94 4.88</td>
<td>5.16 12.29</td>
</tr>
<tr>
<td>4</td>
<td>29/32</td>
<td>45%</td>
<td>100% 81%</td>
<td>2.17 3.27</td>
<td>3.64 5.65</td>
<td>5.18 7.34</td>
</tr>
<tr>
<td>5</td>
<td>33/40</td>
<td>44%</td>
<td>100% 86%</td>
<td>1.56 4.24</td>
<td>2.46 5.81</td>
<td>4.36 9.44</td>
</tr>
<tr>
<td>6</td>
<td>44/45</td>
<td>40%</td>
<td>100% 90%</td>
<td>1.47 2.96</td>
<td>3.14 4.17</td>
<td>4.96 5.82</td>
</tr>
<tr>
<td>7</td>
<td>37/35</td>
<td>39%</td>
<td>100% 83%</td>
<td>1.19 3.05</td>
<td>3.88 5.96</td>
<td>2.96 4.76</td>
</tr>
<tr>
<td>8</td>
<td>33/34</td>
<td>39%</td>
<td>100% 90%</td>
<td>2.18 2.93</td>
<td>3.14 5.84</td>
<td>2.84 9.64</td>
</tr>
<tr>
<td>9</td>
<td>25/28</td>
<td>31%</td>
<td>100% 98%</td>
<td>1.65 3.77</td>
<td>4.43 5.93</td>
<td>2.35 4.74</td>
</tr>
<tr>
<td>10</td>
<td>18/21</td>
<td>27%</td>
<td>100% 82%</td>
<td>2.27 3.79</td>
<td>4.82 5.97</td>
<td>2.35 6.68</td>
</tr>
<tr>
<td>Average</td>
<td>31%</td>
<td>100% 87%</td>
<td>1.6 3.4</td>
<td>3.5 5.2</td>
<td>4.2 7.7</td>
<td></td>
</tr>
</tbody>
</table>

¹ Future Critical Control Application ² Response time in ms. ³ 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](image)

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

![Architecture Model](image)
**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)

<table>
<thead>
<tr>
<th></th>
<th>NOEXT</th>
<th>EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum unattested time</td>
<td>425</td>
<td>150</td>
</tr>
<tr>
<td>Worst-case edge-app response time</td>
<td>350</td>
<td>233</td>
</tr>
<tr>
<td>Average-case edge-app response time</td>
<td>66,421</td>
<td>45,117</td>
</tr>
<tr>
<td>Total critical app latency</td>
<td>4,668</td>
<td>5,141</td>
</tr>
</tbody>
</table>

Comparison of NOEXT vs. EXT

<table>
<thead>
<tr>
<th></th>
<th>NOEXT</th>
<th>EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>121.75</td>
<td>79.75 (-34.5%)</td>
</tr>
<tr>
<td>E2</td>
<td>180.83</td>
<td>107.83 (-40.37%)</td>
</tr>
<tr>
<td>E3</td>
<td>215</td>
<td>185.17 (-13.87%)</td>
</tr>
<tr>
<td>E4</td>
<td>166.25</td>
<td>148.33 (-10.78%)</td>
</tr>
<tr>
<td>E5</td>
<td>161.67</td>
<td>134.67 (-16.7%)</td>
</tr>
<tr>
<td>E6</td>
<td>109.83</td>
<td>109 (-0.76%)</td>
</tr>
<tr>
<td>E7</td>
<td>101.5</td>
<td>65.33 (-35.64%)</td>
</tr>
</tbody>
</table>

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)
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
Research output and overall evaluation

<table>
<thead>
<tr>
<th>Research contribution</th>
<th>Increased security</th>
<th>Increased latency of critical control</th>
<th>Increased availability of hardware, configuration and software management tools</th>
<th>Increased access to real-time data analysis</th>
<th>Reduced time-to-market for new industrial applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK synchronization for virtualization (T)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TASK synchronization scheme (T)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Time-triggered hypervisor (T)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TSN configuration optimization (T)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Clock synchronization for virtualization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Time-triggered networking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wireless networking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TSN configuration optimization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Orchestration method</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Routing configuration optimization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Control virtualization method</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extensible configuration optimization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Application migration method</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scheduling algorithm for elastic applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fault recovery mechanism</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fault-tolerant architecture</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Authentication method</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Intrusion detection method</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fault detection mechanism in TSN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Safety services</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Decomposed deep training solution</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data analytics solution</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The public prototypes are at: http://www.fora-eten.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

IT-OT gap crossed with Dependable Edge Computing

Convergence of IT and OT needed for Industry 4.0

- Enterprise
- Manufacturing
- Machine
- Control

IT

OT

1. Machine
2. Control
3. Manufacturing
4. Enterprise