Execution Environment for Mixed-Criticality Train Applications based on an Integrated Architecture

Hongjie Fang, Roman Obermaisser

University of Siegen, Germany

Abstract— **In the avionic domain, the development trend has moved from federated to integrated architectures, which is called Integrated Modular Avionics (IMA). The ARINC 653 standard defines the execution environment for hosting several avionic software functions within a single computing module using strict temporal and spatial partitioning. ARINC 653 was successfully implemented (e.g., Airbus A380) and achieved its primary goals (cost and weight reduction, enabling modular certification, etc.). However, in the railway domain, an execution environment which enables integration of mixed-criticality Train Control and Monitoring System (TCMS) applications on an integrated architecture is an open research problem. Specific requirements of the railway domain (e.g., inauguration) are not covered by IMA due to the specific relevance in railway systems.**

In this paper, we will analyze the State-of-the-Art (SOTA) of control and monitoring systems in avionic and railway industries and identify the major gaps between these technologies. Based on the analysis, we will put forward our execution environment concept for an integrated architecture in TCMS.

I. INTRODUCTION

In several domains such as the avionic and automotive industry, research on integrating functions with different criticality levels on the same computing platform was carried out and corresponding industry standards (e.g. ARINC 653, AUTOSAR) were released. For example, functions of class A and class B (according to DO-178C [6]) can be integrated on the same APplication/EXecutive (APEX) interface that is defined in ARINC 653 [14] without interfering with each other. The foundations for this integration are mechanisms for temporal and spatial partitioning [14], which are designed to prevent fault propagation between partitions and guarantee the reserved computing resources for each partition, so that safety critical functions cannot be affected by functions of lower criticality.

For another perspective, partitioning is one of the necessary prerequisites for modular certification [10]. Modular certification enables that each function could be certified to the respective criticality level (e.g. from SIL1 to SIL4 according to EN ISO/IEC 61508 [3]). Further more, updates of existing functions require only the recertification of the changed functions due to the segmentation of the application subsystems.

In the railway domain, the TCMS is a train-borne distributed control system which provides a single point of monitoring and controlling of all subsystems within a train. Due to the specific industry constraints and the long development life-cycles, railway transportation systems have suffered from a limited adoption of novel technological advancements in electronic hardware and software, communication networks and embedded computing.

Train inauguration consists of operations executed in case of train composition to give all nodes of the train backbone their addresses, orientation and information about all named nodes on the same backbone [11]. Train inauguration represents a major challenge to the adoption of an integrated architecture, because in other industry domains (e.g., avionic and automotive), there is no consideration of dynamic coupling of components to build up a new system. Existing mechanisms to deal with train inauguration in the railway domain are based on federated architectures, where the train coupling is handled on the train backbone layer. Inauguration in an integrated architecture is an important challenge for the network and the application execution environment.

Ethernet technologies have been successfully adopted in many application areas including train communication networks. The evolution of industrial applications has lead to requirements for more bandwidth, low latencies and temporal predictability. However, the networks based on conventional Ethernet have no capabilities to guarantee bounded latency and minimum jitter for hard real-time applications. In order to provide deterministic network transmissions, the IEEE Time Sensitive Networking (TSN) task group has introduced several extension protocols to the IEEE 802.1 Ethernet standard. TSN enables standard Ethernet infrastructures to converge both critical and non-critical traffic. Motivated by the development of these Ethernet technologies, this paper will concentrate on the adoption of TSN capable networks in the TCMS execution environment.

The contributions of our proposed TCMS execution environment are as following:

- TCMS Execution Environment based on an Integrated Architecture: Our proposed execution environment concept leverages an integrated architecture to achieve cost and weight reduction in the railway domain.
- Support for Mixed-Criticality Applications: The proposed architecture implements temporal and spatial partitioning in order to eliminate the interferences between applications with different safety assurance levels.
- Proposed a Solution for Train Inauguration: Our design leverages existing mechanisms in the railway domain (e.g., TTDB, TCN-URI, TCN-DNS server, etc.) to come up with a solution for the train inauguration challenge.

In this paper, we analyze the prevalent execution environment defined by ARINC 653 for avionic applications based

on IMA in Section II and the execution environment used in today's TCMS in Section III. In Section IV, we identify the gaps between the analyzed execution environments. Based on the analysis, a new execution environment is proposed in Section V to close the identified gaps. The focus of Section VI is the analysis of Commercial Off-The-Shelf (COTS) Real-time Operating Systems (RTOS) and hypervisors to be used in the proof-of-concept. Section VI discusses the conclusion and future research.

II. ARINC 653

This section discusses ARINC 653 [14] which defines the interface between avionic applications and the operating system, in order to analyze the suitability of ARINC 653 for the execution environment in train systems.

A. Temporal Partitioning

Based on ARINC 653 [14], applications residing within an integrated module are partitioned with respect to the execution time, which is called temporal partitioning. All partitions within the same module are scheduled on a fixed and cyclic basis. In order to implement this scheduling mechanism, CPU time is divided into major time frames (MTF) with pre-configured duration. Partitions are activated within at least one time window of every MTF and only one partition can be activated during one time window. The system integrator defines the order of the activated partitions, taking the partition attributes like period and Worst Case Execution Time (WCET) into account.

However, OS overhead like communication acknowledgements and time-outs can also affect the allocated duration of time windows for a specific partition [12]. The possible OS overhead which can affect temporal partitioning should be bounded and known from the viewpoint of the partitions that can be preempted.

From the viewpoint of a partition, it consists of several processes, which can be executed concurrently to achieve requirements like bounded latency. Priority-based preemptive scheduling is a common scheduling technique for processes within the same partition. Preemption locks can also be used by a process to avoid preemption by other processes, when this process is executing a critical section or accessing shared resources within the partition. According to [7], lock preemption can also be adjusted to accommodate a Symmetric Multi-Processing (SMP) configuration on multi-core processor. The impact scope of preemption locks is the scheduling of processes within a partition.

According to ARINC 653, the scheduling of partitions is statically defined in the system configuration file and it stays unchanged during the whole runtime. Lhachemi, Hugo, et al. [17] have developed a model for simultaneous partition allocation and schedule design to automatically adjust application distribution over partitions and scheduling patterns to optimize system performance. Joaquim Rosa, et al. [5] developed an algorithm to provide on-line rescheduling of partitions based on the provided pre-configured scheduling mechanisms called Partition Scheduling Table (PST).

B. Spatial Partitioning

In ARINC 653 [14], partitions have their own allocated memory areas that are statically pre-configured before the system starts up. The allocation of the system memory is based on the partition requirements and it is achieved at the partition level. Processes within the same partition have access to the memory of the partition. Due to the concurrent execution of processes within the same partition, conflicts of accessing the same memory could happen. Therefore, semaphores and mutexes defined in this standard are supposed to provide either concurrent or exclusive access to the partition resources.

In many COTS products (e.g., PikeOS, VxWorks, etc.), a Memory Management Unit (MMU) is used to implement the spatial isolation of memory resources at the hardware level [10].

C. Communication

The defined communication within ARINC 653 can be classified into inter-partition and intra-partition communication.

1) Inter-Partition Communication: Inter-partition communication includes both the communication between partitions residing on the same module and on different modules. Standardization of inter-partition communication enables partitions to be reusable and portable.

Messages are the defined mechanism to convey the communicated information between partitions and they are sent through channels (see Figure 1), each of which links two partitions via ports. A channel is the logical link between partitions and its implementation depends on the underlying platform. Sampling and queuing ports are used to connect channels to partitions and they are statically configured during system integration.

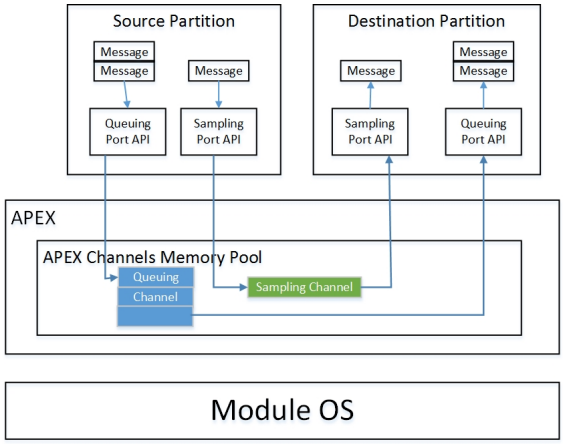

Fig. 1. Scheme of Channel

2) Intra-Partition Communication: Buffers, blackboards, semaphores, events and mutexes are defined to conduct the intra-partition communication and synchronization of processes. Buffers and blackboards provide mainly inter-process communication mechanisms, but they can also be

used to synchronize processes. Semaphores and mutexes are used to synchronize processes which access the same memory resources. Events are defined for one process to trigger another process.

III. Execution Environments in Train Control and Monitoring Systems (TCMS)

This section discusses today's execution environments in the railway domain. Central points are communication interfaces based on Train Communication Networks (TCN), train inauguration, distributed applications in train systems and functional addressing.

A. Train Communication Network (TCN)

TCMS consists mainly of hardware platforms, software execution environment, operating interfaces, I/O devices and underlying data networks. The data network in the railway domain is defined as the Train Communication Network (TCN) according to IEC 61375 [8]. TCN is different from a normal LAN due to the free coupling of vehicles which compose the whole train [4].

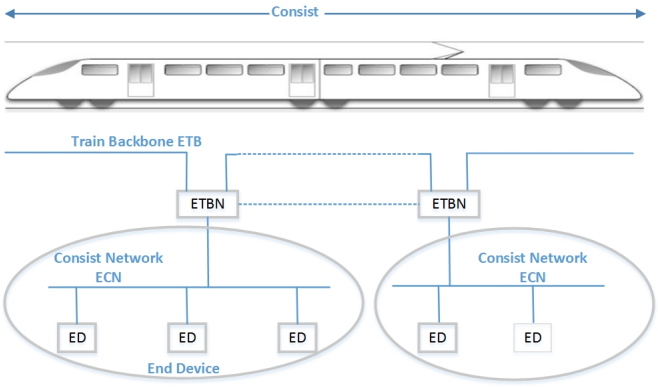

Fig. 2. The hierarchical architecture of TCN

According to the hierarchical architecture of TCN shown in Figure 2, TCN defines a two level network architecture with the Train Backbone (TB) and the Consist Network (CN). In our research, we consider the TCN using Ethernet technology that results in the Ethernet Train Backbone (ETB) and Ethernet Consist Network (ECN). An ETB Node (ETBN) is used to connect the ETB with an ECN.

B. Train Inauguration

The action consisting in configuring the train network is called train inauguration [11]. Vehicles are free to couple and decouple with each other, thereby causing the process of inauguration. The coupling of vehicles is used to connect rolling stocks to form a whole train. For example, the train in Figure 3 is composed of three cars, each of which has one consist network and can be treated as an independent consist. The connecting of two consists at the ETB level in Figure 3 results in the train inauguration process.

From the view point of the whole train, inauguration only affects the ETB network, while ECN is always unchanged during inauguration. The ETBN is responsible for the inauguration process using the Train Topology Discovery Protocol (TTDP) [11]. The TTDP is designed to build the physical and the logical topology of the train. The physical topology of a train is represented by an ordered and oriented list of ETBNs. The logical topology of a train is defined by an ordered and oriented list of train subnets.

After inauguration, all the information related to the actual train composition and the actual ETB state will be stored in the Train Topology Database (TTDB) [13]. In each consist, there must be at least one instance of the TTDB and the instance resides normally on the ETBN [13]. The TTDB is maintained by a TTDB manager, which is designed to manipulate the TTDB and provide an interface for retrieving information (e.g., train direction, consist ID, functions of consists, etc.) of the whole train [13].

C. Distributed Train Applications

The TCN architecture introduced in IEC 61375-1 [8] defines the basic framework for distributed applications at the train level. An example of a distributed application is illustrated in Figure 3, which originates from IEC 61375-2-4 standard [16].

Based on the architecture of the distributed application in Figure 3, a distributed function consists of a function leader, function followers and function devices. The function leader is responsible for sending out commands to function followers and collecting the reactions from function followers. A function follower receives instructions from the function leader and stimulates the function devices within the same consist as well as transfers the feedback from these function devices to the function leader. A function device simply executes the commands from function followers and provides the results back to the function follower.

In this example, communication between a function leader and function followers could require inter-consist communication. The connections between a function follower and the function devices are always within the same consist.

D. Functional Addressing

Since applications (e.g., door control) could be distributed widely in the whole train, the problem of addressing a functional device (e.g., function leader, function follower) must be resolved. Basically, the TCN URI is introduced in IEC 61375-2-3 for the functional addressing [13] for this communication purpose.

1) TCN URI: Applications within TCMS are addressed using functional names based on TCN domain names. These names are defined on top of IP and can be treated as the domain name system of the Internet [13]. The TCN domain name space is defined by a tree structure with the local train being the root. A function can be addressed using a single functional name, while it could have different IP addresses in different consists. Problems caused by changing IP addresses of functional entities are resolved by the TCN DNS server and there is no impact on the implementation of applications.

All communication entities are identified by fully qualified TCN domain names that are derived from the TCN

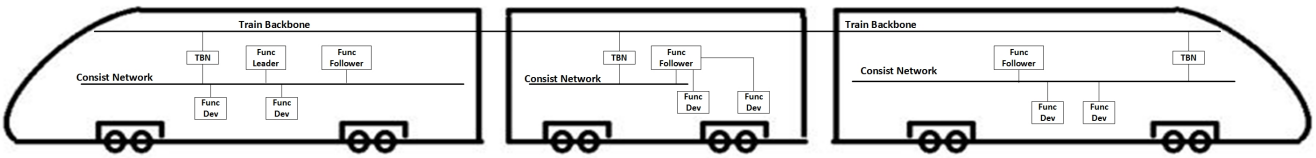

Fig. 3. Architecture of distributed applications

domain name space. The name space is defined by an URI scheme following the recommendation of RFC 3986 [13]. A TCN URI is used to identify either a source or destination function/device of communication data and there are also reserved TCN URIs for specific functions.

2) TCN DNS: The TCN DNS server provides the service for resolving TCN URIs to IP addresses and according to IEC 61375-2-3 [13], there exists at least one TCN DNS server for functional address resolution within a consist network. For some reserved TCN URIs, their IP addresses are predefined and there is no need to refer to the DNS server for TCN URI resolving. The TCN DNS server gets access to the local TTDB (mostly residing on the ETBN) via the TTDB manager to retrieve the up-to-date information of the whole train and computes the IP addresses using the pre-defined rules.

IV. Gaps between ARINC 653 and TCMS

Since ARINC 653 was developed originally for the avionic domain, inauguration is not considered in this standard. This section focuses on discussion of the gaps between execution environments in avionic and railway domains. For TCMS based on an integrated architecture, inauguration is an important gap to be closed. Inauguration needs to be handled both in the network and in the software execution environment.

A. Execution Environment

Existing TCMS execution environments are based on a federated architecture which is different to an integrated architecture used in ARINC 653. Every node in a federated system hosts one function, therefore, the application execution environment on a node is function-specific. In integrated systems, one computing node can host multiple applications, because the execution environment and the underlying platform provide the capability to spatially and temporally separate different functions.

The dynamic reconfiguration of today's TCMS execution environments is achieved by the inauguration process. In the ARINC 653 standard, the configuration of the execution environment is done by the system integrator during design time. For example, configurations of intra-partition and inter-partition communication are static. Reconfiguration of the execution environment is often reduced to switching between predefined configuration modes, which is mainly due to safety concerns, because pre-configured scheduling schemes can be statically analyzed and proven to be safe.

B. Communication Network

Regarding the architecture of communication network, TCN was defined as a two-level architecture (i.e., train backbone and consist network). In this architecture, the inauguration process is carried out at the train backbone level and consist network can be statically configured. In avionic domain, ARINC 664 Part 7 [1] defines the aircraft data network Avionics Full Duplex Switched Ethernet (AFDX) for the IMA architecture. AFDX emulates the point-to-point connectivity via Virtual Links (VL) to obtain a deterministic network for avionic applications. The cascaded star topology is used for AFDX and the coupling of two existing networks can cause the reconfiguration of the whole data network.

As discussed in Section III, in the railway domain, ETBNs are assigned to run the TTDP to update the train-wide information in the TTDB. Applications are not aware of the topology changes, since TCN DNS servers address the dynamic addressing on the network layer. In ARINC 653 and ARINC 664, the system integrator statically configures the topology of the network. Therefore, there are no repositories containing information of physical and logical topologies in avionic systems. Applications should be reconfigured after topology changes.

C. Temporal and Spatial Partitioning

In the railway domain, applications of TCMS can have different safety criticality levels (e.g., SIL1 to SIL4). The execution environment based on IMA architecture defined in ARINC 653 provides shared computing resources for different applications. The existing execution environments in railway systems achieve partitioning due to the natural separation of nodes in federated systems. In order to prevent interference between applications with different criticality levels and guarantee sufficient hardware resources for applications, the TCMS execution environment based on an integrated architecture should ensure spatial partitioning for the hosted applications. One possible implementation of spatial partitioning is hardware MMU based.

Some of the TCMS functions are hard real time control functions where missing the deadline of such functions can cause serious consequences. For example, when a brake control function misses its deadline and fails to stimulate the function actor, the train does not brake and this can result in passenger injury or even death. The existing distributed TCMS functions do not share computing resources due to the nature of a federated architecture. In order to guarantee

the real-time property of the TCMS system on an integrated architecture, the execution environment should ensure strict system-level time partitioning. Similar to ARINC 653, partitions should be scheduled on a cyclic basis, which enables the execution environment to allocate computing resources for each function to meet the timing requirements.

D. Reconfiguration

In railway systems, coupling of vehicles is flexible due to the different choices of vehicle types and the length of the whole train. In the avionic domain, aircrafts are not required to couple with each other at the data network level. Due to the flexibility of train coupling and decoupling, the TCN URIs are defined to address functional entities. The TCN DNS server is used to resolve a TCN URI to IP addresses of communicating entities. In ARINC 664, end-to-end communications are identified by a quintuplet of source UDP port, source IP, VL identification (i.e., MAC address), destination IP and destination UDP port [1]. The partition ports defined in ARINC 653 are mapped to the communication ports defined in ARINC 664. Therefore, communications in IMA systems are statically configured and should be extended for TCMS based on an integrated architecture.

Due to the nature of train inauguration and since IPv4 is now widely used in the railway domain, after lengthening or shortening a train, the IP addresses of logical or physical entities within a consist will be reassigned on the train backbone layer. The execution environment based on an integrated architecture should provide the capability to address the dynamic attributes caused by train inauguration and provide static APIs for the hosted applications (i.e., applications do not need to be adjusted after inauguration).

V. Execution Environment For Integrated TCMS-Architecture

In this section, the TCMS application execution environment based on an integrated architecture will be shown and details of closing gaps between TCMS and ARINC 653 will be addressed.

A. Structure

Figure 4 illustrates the proposed TCMS execution environment for mixed-criticality train applications based on an integrated architecture. The designed execution environment resides on top of TSN capable network[21] and provides a hardware abstraction for the hosted partitions. In order to provide address and location transparency for the hosted partitions, the execution environment needs to provide the functionalities of switching messages within the same module or cross the module boundary, while the partitions utilize the same communication interfaces. The VTSN defines the logical time sensitive links between partitions. Partitions communicate with each other either directly via the virtual TSN (VTSN) which is implemented in the scope of the execution environment, or via the proxy partition. Proxy partition can transport those messages, for which there exist no logical links.

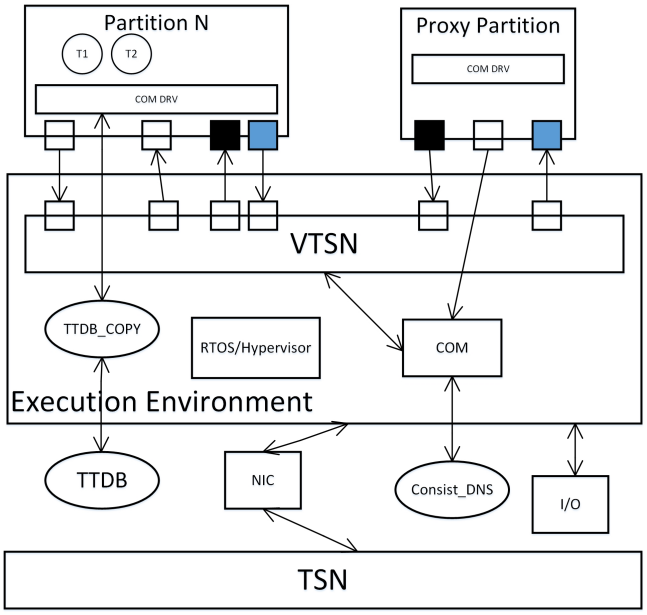

Fig. 4. TCMS Application Execution Environment

B. Conceptional Details

In this section, the proposed concepts will be illustrated in detail.

1) Data Network: In our designed architecture, the underlying data network is TSN [21]. TSN enables packet transport with bounded low latency, low delay and low packet loss rate over Ethernet. Bounded latency for scheduled traffic through switch network meets strict deterministic requirements of control applications (e.g., in automotive and industrial domains). Moreover, time-critical traffic can be sent via a standard Ethernet infrastructure running the TSN protocols [22]. Therefore, a standard Ethernet infrastructure based on TSN can converge both critical and non-critical traffic. For this reason, sub-systems could be integrated in a simple way.

In TCN, ETB and ECN are modified Ethernet variants for train communication systems. Therefore, it can be difficult for a train system to cooperate with other networks at the data network level. However, from the viewpoint of future development, industrial systems are evolving towards an industrial Internet of Things (IoT) [22]. This trend motivates TSN, because TSN is designed to meet both control and connectivity requirements of control applications.

2) Partition: The concept of partition comes from ARINC 653 [14], where applications are partitioned into components which have temporal and spatial segregation between each other. A partition consists of at least one task (e.g., T1 or T2 in Figure 4) and all tasks within the same partition can share access to the partition resources. Scheduling of tasks should be free to be decided by the application developer, priority-based preemptive scheduling is an example strategy at partition level, when the tasks within a partition share the memory and other spatial resources of the partition. Intra-partition communication and synchronization can be done

by implementing similar mechanisms like blackboard, buffer, semaphore, event and mutex which are defined in ARINC 653.

One partition has an important component called Communication Driver (COM DRV) that is responsible for data transportation. COM DRV will be illustrated in the following sections.

3) Execution Environment: Partitions have direct access to the execution environment via well-defined services (e.g., time service, partition management service, etc.). The VTSN presents the logical time sensitive link between partitions and they get access to VTSN through ports. Similar to the communication mechanism in ARINC 653 standard, ports could be classified into sampling and queuing ports, depending on the functional requirements of communicating entities. One partition can simply send a message into a port and the execution environment should deliver the message to the destination while guaranteeing to meet the requirement of latency, jitter, etc.

Due to the reallocation of IP addresses during inauguration process, messages should be location transparent to guarantee the portability of developed partitions. In other words, the messages sent by the partitions contain no IP addresses. Communication destinations are addressed using TCN URI and the COM DRV component of a normal partition is designed to map TCN URIs to ports. In this scenario, COM DRV needs to record the setup of ports within the partition during system initialization time and the corresponding communication partners. Informations about functions residing on the other modules in the same/other consists could be retrieved from the TTDB. Since the TTDB exits normally only on the ETBN, it is of benefit to create a copy of TTDB within every execution environment and keep them synchronized. Another possibility is that COM DRV refers directly to the TTDB in the same consist, which may lead to concurrent access of different partitions to the TTDB.

Ports could be mapped to a shared memory pool within the execution environment and the COM component can either map the destination port of the same module to the same memory address or send messages to the data network (i.e., TSN in our design). The COM component refers to the DNS server to retrieve the destination IP address of a message. The destination IP address could be used by COM to identify whether the destination partition resides on the same module or not. In the later case, the COM component packs the necessary information (e.g., header, CRC, etc.) to the message (e.g., using TRDP) and sends it out through Network Interface (NIC). If the TTDB is saved in every execution environment, it is of benefit to implement the DNS server in the execution environment locally, since the DNS server needs to retrieve the train topology information from the TTDB and in this way, the underlying data network will be released from the communication between COM DRV and TTDB, between COM and DNS server as well as between DNS server and TTDB. But it also brings overhead for the local TTDB to be synchronized with the one on ETBN.

4) Inter-Partition Communication: In the designed TCMS execution environment, inter-partition communication is conducted by a proxy partition and the VTSN which is connected to partitions via ports. The COM DRV within one normal partition should map messages to pre-defined ports. In case of mis-mapping of a destination TCN URI, the message will be simply sent to the blue port in Figure 4 that is directly linked to the proxy partition. The COM DRV in the proxy partition should be able to resolve the destination URI to the IP address. If the destination of the message resides on the same module, then the proxy partition sends out the message via the black port which is defined as a multi-casting port to all the partitions within the same module. In the case that the destination IP is on another module, the message will be sent out through the COM component to the underlying TSN.

Ports are considered to be defined statically during system initialization in order to reduce the complexity of inauguration management in the execution environment. The later discovered functions without pre-allocated ports will be communicated via the proxy partition. The new integrated functions with allocated ports will be discovered and recorded in the TTDB which provides the train-wide information for COM DRV within partitions, the COM components in the execution environments and the DNS server. These components will merge the connections of the discovered functions to the existing VTSN and finish the train inauguration within the execution environment.

As discussed in the Section of the data network, TSN is chosen as the data network candidate. The VTSN is designed in the execution environment based on the following arguments.

- In order to abstract the underlying data network for the partitions, VTSN is designed to provide the communication interface of the execution environment. In this way, the data transportation is transparent for the partitions.
- VTSN is designed for the inter-partition communication within the same execution environment, because TSN is used to connect different computing platforms and the execution environment needs to provide the capability of inter-partition communication within the same computing module.
- Since partitions share the physical TSN network, for communications between partitions residing in different execution environments, VTSN and the COM component should enable the traffic shaping and provide the gateway to a physical TSN.

As discussed in Section II, channels are defined in ARINC 653 as the basic mechanism for the communication between partitions. From the viewpoint of APEX, channels are statically configured. For the designed TCMS execution environment, VTSN can adjust to train topology changes. For example, when a new partition running a brake function is discovered and recorded in TTDB, VTSN can include this new partition as a destination as soon as a multi-cast message for a brake function arrives. The VTSN can be treated as an

instantiation of a channel with the extended capabilities to deal with inauguration in train systems.

VI. COTS RTOS and Hypervisor

This section focuses on the analysis of COTS RTOS and hypervisors, in order to identify the suitable candidate for the proof-of-concept.

For the COTS RTOS, a detail analysis of VxWorks [23][15], LynxOS [15] and Integrity [18] has been carried out. All these three RTOSes support hard real-time capability, which means their scheduling mechanisms are designed to guarantee computing resources for applications to meet their deadlines. The scheduling strategy of VxWorks supports changing the scheduling scheme via changing task priorities. In the aspect of memory management, LynxOS guarantees spatial partitioning through an MMU but the assigned memory of a partition is unchangeable on the fly. Integrity and VxWorks use either a memory protection unit (MPU) or memory management services to implement spatial partitioning.

PikeOS [20] and XtratuM [2] are the analyzed hypervisors. PikeOS is a para-virtualized hypervisor and XtratuM is a bare-metal hypervisor [9]. PikeOS implements a combination of time and priority driven scheduling, while XtratuM schedules partitions based on a fixed cyclic scheme. For these two hypervisors, switching between pre-configured scheduling schemes at runtime is possible. They both choose an MMU to guarantee spatial isolation of partitions. In the implementation of PikeOS, there is a background partition which contains both high priority tasks and low priority tasks. These tasks in the background partition are designed either to preempt tasks in foreground partitions or to consume unused time slices of foreground partitions [19].

For our research, we intend to implement our designed framework based on a hypervisor, because a hypervisor provides virtulization of the underlying hardware and is also able to host different RTOSes (e.g., Linux, OSEK, etc.) as well as provides different APIs (e.g., ARINC 653, POSIX, etc.). Our future prof-of-concept of the designed TCMS execution environment will be carried out based on PikeOS, since PikeOS provides a more flexible scheduling scheme compared to XtratuM.

VII. Conclusion and future research

This paper analyzes the state-of-the-art of the execution environments in the railway domain, which follow a federated architecture paradigm and are thus not cost effective. The state-of-the-art of execution environments in the avionic domain including ARINC 653 and its industrial successful implementation (e.g., Airbus A380) motivate research for mapping these concepts to the railway domain. The most important difference between avionic and railway applications is the inauguration problem. After identifying the main gaps, we proposed the architecture of the TCMS application execution environment based on an integrated architecture. Our designed execution environment implements the temporal and spatial partitioning concepts from ARINC 653 and proposes a solution for the inauguration problem based on the existing mechanisms in the railway domain (e.g., TTDB, DNS server, etc.).

ACKNOWLEDGMENT

Research reported in this paper is funded by Safe4RAIL Project, Grant Agreement No. 730830.

References

[1] Aircraft Data Network Part 7 Avionics Full Duplex Switched Ethernet AFDX) Network, Airlines Electronic Engineering Committee Std., 2005.

[2] Crespo, Alfons, Ismael Ripoll, and Miguel Masmano. "Partitioned embedded architecture based on hypervisor: The xtratum approach." Dependable Computing Conference (EDCC), 2010 European. IEEE, 2010.

[3] IEC 61508 Edition 2.0: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, IEC: Int. Electrotechnical Commission, 2010.

[4] Zeng, Zeng, et al. "A distributed comparison algorithm for train inauguration protocols over ethernet." E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on. IEEE, 2010.

[5] Rosa, Joaquim, Joao Craveiro, and Jos Rufino. "Safe online reconfiguration of time-and space-partitioned systems." Industrial Informatics (INDIN), 2011 9th IEEE International Conference on. IEEE, 2011.

[6] DO-178C/ED-12C, "Software Considerations in Airborne Systems and Equipment Certification", published by RTCA and EUROCAE, 2012.

[7] Huyck, Patrick. "ARINC 653 and multi-core microprocessorsConsiderations and potential impacts." Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st. IEEE, 2012.

[8] IEC 61375-1:2012. Train communication network (TCN) - part 1: TCN general architecture

[9] Sandstrm, Kristian, et al. "Virtualization technologies in embedded real-time systems." Emerging Technologies and Factory Automation (ETFA), 2013 IEEE 18th Conference on. IEEE, 2013.

[10] Obermaisser, Roman, and Donatus Weber. "Architectures for mixed-criticality systems based on networked multi-core chips." Emerging Technology and Factory Automation (ETFA), 2014 IEEE. IEEE, 2014.

[11] IEC 61375-2-5:2014. Electronic railway equipment - train communication network (TCN) - part 2-5: Ethernet train backbone.

[12] Arcaro, Luis Fernando, and Rmulo Silva de Oliveira. "Lessons learned from the development of an ARINC 653 compatible Operating System." Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on. IEEE, 2015

[13] IEC 61375-2-3:2015. Electronic railway equipment Train communication network (TCN) Part 2-3: TCN communication profile

[14] Airlines Electronic Engineering Committee. "Avionics application software standard interface part 1-required services." ARINC Document ARINC Specification 653P1-4, Aeronautical Radio, Inc., Maryland (2015).

[15] Mahani, Reihaneh Torkzadeh, and Negin Mahani. "VxWorks vs. LynxOS Real-Time Operating Systems for Embedded Systems." (2015).

[16] IEC 61375-2-4:2016. Electronic railway equipment Train communication network (TCN) Part 2-4: TCN - Train Communication Network - Application Profile

[17] Lhachemi, Hugo, et al. "Partition modeling and optimization of ARINC 653 operating systems in the context of IMA." Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th. IEEE, 2016.

[18] Integrity RTOS, Green Hills, [Online]. Available: http://www.ghs.com/products/rtos/integrity.html.

[19] SYSGO Whitepaper, Combining Partitioning and Virtualization for Safety-Critical Systems

[20] SYSGO Whitepaper, PikeOS Safe Real-Time Scheduling, Adaptive Time-Partitioning Scheduler for EN 50128 certified Multi-Core Platforms

[21] Time-Sensitive Networking Task Group, IEEE, [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[22] TTTech White Paper, Time-Sensitive Networking (TSN)

[23] Wind River White Paper, Safety-Critical Software Development for Integrated Modular Avionic