Towards an Evaluation Environment for Digital Interlocking Networking

Frederic Reiter[1], Lukas Iffländer[1], Ulrich Maschek[2], Richard Kahl [2]

1 Deutsches Zentrum für Schienenverkehrsforschung beim Eisenbahn-Bundesamt

2 Technische Universität Dresden, Professur für Verkehrssicherungstechnik

1 Introduction

Interlocking faces a major restructuring. Existing heterogenic systems require massive maintenance structures able to support four different generations of interlocking (mechanical, electromechanical, relay and electronic interlocking) but also various types from various manufacturers for each generation. These types rarely share replacement parts and are usually incompatible with each other. Connections between neighboring interlockings require proprietary adapters. Furthermore, once interlocking for a station is commissioned, replacement parts and modifications can only be acquired from the original supplier creating an undesired supply monopoly.

Germany's incumbent infrastructure manager Deutsche Bahn, therefore, decided to switch to a new type of interlocking, termed digital interlocking. While one might call the decision process in electronic interlocking – and generally all states in interlocking systems – digital, the differences to "classical" electronic interlocking is that field elements are connected to the control unit using standardized data busses, protocols and interfaces. Through this standardization, infrastructure managers can exchange field elements (e.g., light signals, switches, axle-counters) from one manufacturer with elements from a competitor, eliminating dependencies. Furthermore, systems can easily be upgraded or exchanged without replacing the remaining systems – today, when replacing an electronic interlocking it is often necessary to also replace all field elements.

Germany intends to digitize its state-owned infrastructure's interlocking systems by 2040. The German railway industry lobbies for an even more ambitious timeline of 2035. Over a hundred thousand field elements require a migration from classical interlocking to digital interlocking. This goal is ambitious not only when it comes to planning processes but also, when it comes to testing and authorization for the used software.

Current approaches to integration testing do not support the planned implementation speed. Deutsche Bahn creates expensive and complicated test environments for digital interlocking projects like the line section between Mertingen and Meitingen. These environments comprise an exact clone of the interlocking except for the physical field elements. Neither is continuing this approach for a large-scale rollout economically feasible,

[1] Korrespondierende Autoren: reiterf@dzsf.bund.de, ifflaenderl@dzsf.bund.de

nor does the necessary amount of personnel exist to attain the desired speed. With our contribution in this paper, we want to support the transition from real-world environments to their virtualized equivalents.

Additionally, to the requirement for faster testing, the need for faster interlocking arises. Studies found that the interlocking operation speed has a significant impact on the capacity of stations and routes [1]. Therefore, it is necessary to identify potentials to further shorten round setting times.

In this work, we propose and describe a design for a fully virtualized evaluation environment that reduces complexity while keeping close to reality. To our knowledge, similar research with a focus on digital interlocking systems has not been published with the exception of [8], which examines interlocking interfaces qualitatively but provides no basis for quantitative evaluation of performance. Our environment allows to perform feature and performance testing for all steps before the integration with the physical field elements. Furthermore, the system is designed to incorporate hardware-in-the-loop testing, allowing to add real-life field elements or interlocking into the environment while keeping the remaining environment virtualized.

Our environment uses virtualized machines to create an exact replica of the rail IP System (bbIP) specified by Deutsche Bahn. Field elements and interlocking communicate using the Rail Safe Transport Application (RaSTA) protocol [9]. As application layer protocol, we use the EULYNX specification that builds upon RaSTA [15]. EULYNX is a European initiative by 14 infrastructure managers to standardize interfaces and elements of the signaling systems.

The proposed implementation reaches prototype level and comprises a simplified interlocking, the network between interlocking and field elements (including a redundant connection as specified for RaSTA), sample field elements (switches and light signals) and monitoring functionality. This functionality allows to monitor and benchmark the entire process of the route setting time at every step to identify which parts of the system take up what amount of time.

We evaluate our system regarding the ability to test and verify the functionality of software components and to measure the performance and the theoretically possible route setting times. For the first goal, we found multiple issues in an existing open source RaSTA implementation and validated the correction inside our environment. For the route setting time, we found, that the protocol aspect is nearly negligible and, in theory, significantly shorter route resolution time and route setting times than required by Deutsche Bahn are possible. This result suggests a requirements reevaluation for further projects.

The remainder of this paper is structured as follows: After this introduction, in Section 2, we introduce technical background on bbIP and modular testing. Section 3 shows our approach to system design and implementation. Afterwards, we evaluate our approach in Section 4 and discuss the results in Section 5. Lastly, Section 6 provides a short summary and an outlook on future work.

2 Background

EULYNX is poised to be the rail industries USB standard with respect to field elements and has been adopted by Deutsche Bahn infrastructure company DB Netz AG as successor to NeuPro – Deutsche Bahn's original standardization proposal [10]. Standardization enables a plethora of benefits, the most pronounced of which are opportunities for small and medium sized companies to enter the market, which makes for cheaper acquisition and maintenance of infrastructure, as well as easier replacement of old parts with the components of a different supplier [6]. EULYNX interfaces are specified in semi-publicly available documents, which determine, next to other things, the contents of the exchanged messages and how and when each component should send which kind of message [15]. It also specifies a safe communication network, which for the DB-operated area, will be the bbIP. Quality-of-Service requirements for the network's latency posed by the EULYNX specification are 50 ms for the Standard Wired Profile.

The bbIP network is made up of three layers, each for a different area of operation. They consist of completely redundant blue and grey network planes. The lowermost layer (access layer) connects the site of operation (TSO), which includes interlocking and Maintenance & Data Management (MDM), to the field element access point, which in turn connects to the field elements' Object Controllers (OC). The OC themselves come with the actual application module and two security components, a crypto-box and a VPN [3].

As mentioned in the opening, the usage of standardized interfaces and commercial-of-the-shelf (COTS) components for command-and-control poses new challenges and opportunities for the process of testing and authorization. Caspar et al. in [5] previously concluded that the interlockings composition of a multitude of modules which originate from different sources would necessarily lead to a change in validation practices, as the conventionally intended test of the system could only be conducted after it had been built in the field. At this time, the validation effort would lead to pronounced losses in time and money, as well as needless delays in project completion. They propose a multi-phased test procedure, called "modular, hierarchical testing", which schedules integration tests of the test unit with an increasing number of interfacing systems and in increasingly realistic environments. An Evaluation environment like ours represents the tool, which is needed to conduct the tests of the intermediate phases in an approach like the one by Caspar et al.

3 Methods

Structurally, a digital interlocking is an arrangement of computers which feature-specific hard- and software, interconnected by the bbIP network which bases on standard IPv4. The behavior of a digital interlocking can be emulated by realizing a representation of such computers in correspondence to the architecture of the target network. By configuring the environment with the respective hardware capabilities and network Quality of Service (QoS) parameters (e. g. maximum throughput, latency, or transmission failure rate), results of experiments in the emulated scenario can approach real world application accuracy. To authentically model the bbIP environment, we requested and received latency measurement data for the newly built TSO Mertingen–Meitingen from DB Netz AG [4]. Even for higher amounts of bytes per packet (1024 bytes) than usually reached by cryptographically secured and RaSTA-headed EULYNX Protocol Data Interface (PDI) messages, the latency didn't surpass 1 ms. We used the respectively measured latency of 0.562 ms for usual packet sizes of 512 bytes in the simulation.

To support a wide range of different use cases, the proposed evaluation environment is based on modern virtualization techniques. Thus, instead of using real hardware for each single machine, we create virtual machines for every component. Virtual machine hypervisors create a hardware-agnostic environment of virtual processors, memory and drives, on which they allow to run entire operating systems without them realizing they do not run on actual hardware. Inside fully virtualized machines, container virtualization allows to run multiple processes on the same machine without them having knowledge of each other or being able to

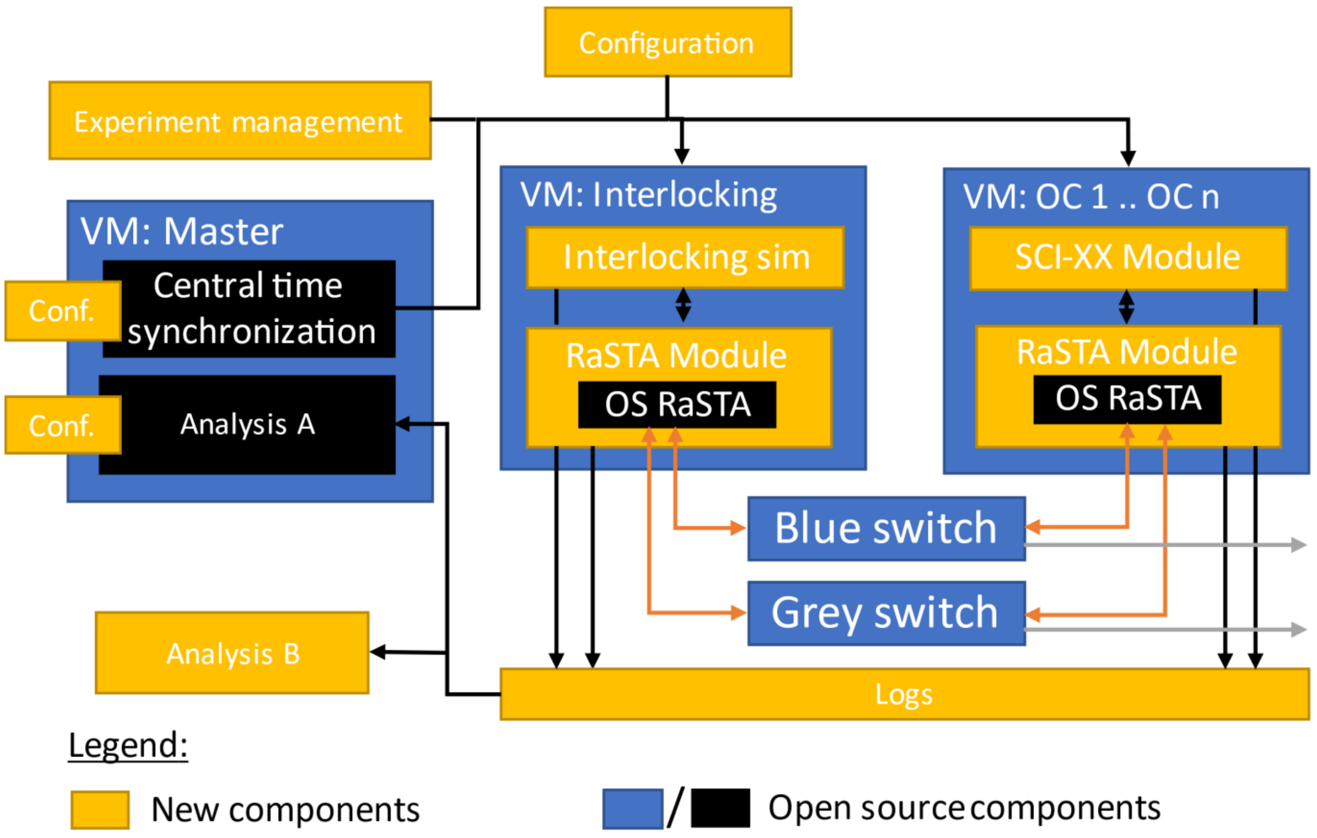

Figure 1: Evaluation environment structure comprising existing (open source) components and specific components for the desired applications.The environment comprises several virtual machines (VMs) to represent the interlocking and the Obect Controllers (OC). OC and interlocking each comprise two containers. One container handles the networking using the RaSTA protocol while the other container is responsible for the functional aspects. OC use the varius standard communciation interfaces (SCI) specified by Eulynx. Black arrows represent the experiment control communication while orange arrows represent interlocking communication. Grey arrows hint at possible extensions to other components like control stations or radio block centers.

interfere. While the separation is less than for full virtualization the omission of having multiple instances of the operating system saves a significant amount of resources.

Figure 1 shows the architecture for our environment called DSTW-Sim. Besides fixed virtual machines for the interlocking and an experiment controller, virtualization enables the deployment of an arbitrary number of field elements. Furthermore, the incorporation of physical elements is possible, but not mandatory, allowing for so called hardware-in-the-loop testing where physical objects are validated and evaluated using simulated or virtualized environments.

The architecture of the bbIP infrastructure is reproduced accurately in DSTW-Sim, meaning every object controller, the interlocking, as well as all switches of the redundant network planes are represented by their own virtual machine (VM). To keep the behavior of the test environment as close as possible to the real system, we use hypervisor-based virtualization provided by **VirtualBox** v6.1.38 [16] for each individual VM. Where applicable, container-based virtualization provided by **Docker** v20.10.12 [17] is used inside the VM to attain a modular design and make for an easy replacement of subcomponents [2].

For provisioning, we use the open source tool **Vagrant** v2.3.4[14] which is configured automatically according to the user-defined setup of the experiment. Vagrant is a tool for building complete environments using virtualized machines. Once configured, the entire evaluation environment can be easily provisioned or destroyed without manual intervention.

A newly designed configuration language based on Yet Another Markup Language (YAML) [13] provides a way for the user to specify the setup of any combination of field elements. The elements are given by category and name. We use a script written in **Python** [18] to translate the experiment configuration to a Vagrant configuration, create all experiment specific files, control the experiment and analyze the results.

The OC in DSTW-Sim consists of two modules: The application module is implemented in the memory-safe language **Rust** [19] and processes PDI messages according to the EULYNX interface specification, while the RaSTA module handles the network communication as intended by EULYNX. The two modules communicate locally via User Datagram Protocol (UDP).

The time and content of all messages to be sent and received by the application layer in the process of route setting is implemented to the exact EULYNX requirements specification to set the conditions for realistic measurement results. Specifically, the message processing features of the Standard Communication Interface for Point (SCI-P), Light Signal (SCI-LS) and electronic interlocking were realized. Simplifications in this initial implementation include omitting the cryptographic components of the OC otherwise used in the bbIP architecture and assuming the PDI connection between interlocking and OC to be already established. Additionally, the simulation of the interlocking supports only the setting of one route at a time.

Crucially, the evaluation environment is executable on every computer that runs a standard operating system (Windows, Linux, MacOS) after installing the open source utilities Vagrant, VirtualBox and Docker. This property yields excellent reproducibility of experiments. We realized all implementations solely by using open source components.

Our environment allows us to instrument various experiments, collect logging information and experiment results and analyze these results. As a method of validating our system, we designed test cases for the network protocol RaSTA or, more specifically, it's open source implementation by Railway-CCS [7]. RaSTA features mechanics to detect the untimely delivery of a message, as well as detection and correction of packet loss. To test these features, we use failure injection by the network emulation software **netem** [11] to simulate degraded QoS connectivity profiles.

While not required for the functionality of the system, we implement internal clock synchronization for all participating computers using the Linux kernel-integrated tool **linuxPTP** [12] for the purpose of comparing logged timestamps across the systems in the environment.

4 Results

We evaluated our environment's functionality by setting up a sample railway station with two light signals and two switches (see Figure 2). We measured the performance of the modelled systems and the RaSTA implementation by repeatedly setting the specified routes in the interlocking simulation and logging the exchange of all relevant messages. The necessary steps for route setting according to the EULYNX specification are pictured in Figure 3. These include all steps of the interlocking domain before a movement authority would be passed to the radio block center (RBC). The steps represented by white boxes are not included in the calculation for the route setting time.

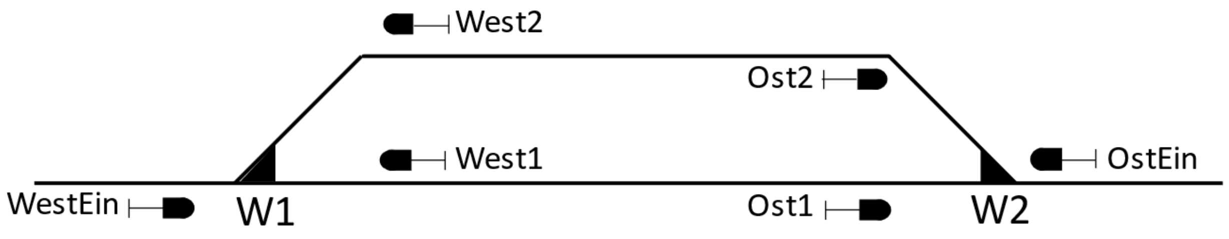

Figure 2: Sample railway station used for this paper comprising six light signals and two points.

EULYNX interaction between all components performed flawlessly. However, we found several issues within the used open source RaSTA implementation. For example, our environment showed implementation errors in RaSTA's check for untimely delivery with respect to the clock independency mechanism that led to unintentionally terminated connections (see Figure 4). Specifically, the system clocks don't have to be synchronized between RaSTA communication partners, because they are never compared with each other. At the time of initialization, the local clocks' time is used as a placeholder value. Erroneously, this initialization happened for every received heartbeat message (HB), making the system race its own clock. The result of this

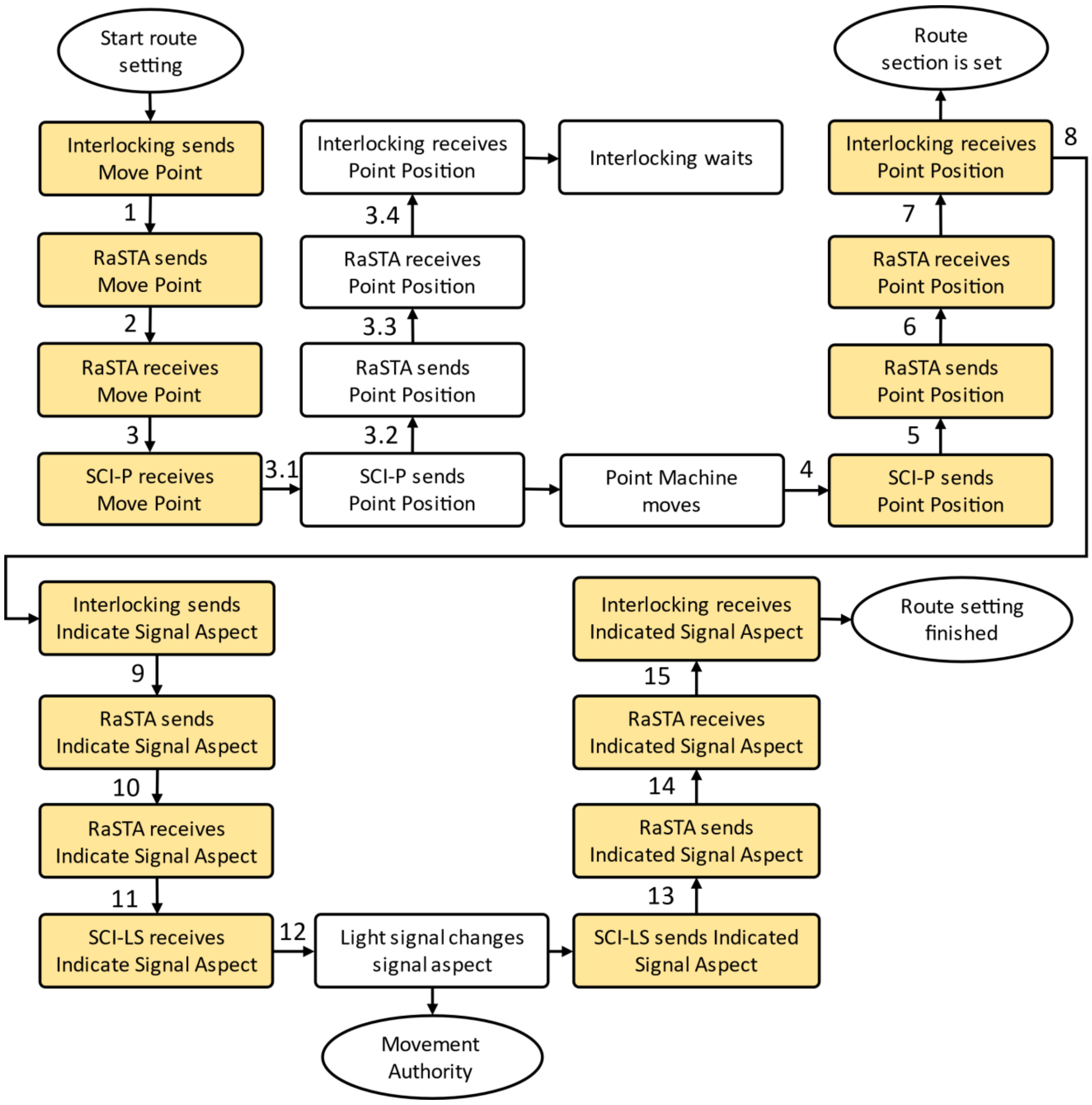
Start route setting
Interlocking sends Move Point
1
RaSTA sends Move Point
2
RaSTA receives Move Point
3
SCI-P receives Move Point
3.1
SCI-P sends Point Position
3.2
RaSTA sends Point Position
3.3
RaSTA receives Point Position
3.4
Interlocking receives Point Position
Interlocking waits
Point Machine moves
4
SCI-P sends Point Position
5
RaSTA sends Point Position
6
RaSTA receives Point Position
7
Interlocking receives Point Position
8
Route section is set
Interlocking sends Indicate Signal Aspect
9
RaSTA sends Indicate Signal Aspect
10
RaSTA receives Indicate Signal Aspect
11
SCI-LS receives Indicate Signal Aspect
12
Light signal changes signal aspect
Movement Authority
SCI-LS sends Indicated Signal Aspect
13
RaSTA sends Indicated Signal Aspect
14
RaSTA receives Indicated Signal Aspect
15
Interlocking receives Indicated Signal Aspect
Route setting finished

Figure 3: Phases of the route setting process with respect to exchanged EULYNX messages.

were messages, that appeared to have been sent even before the previous message which had already confirmed the current timestamp (CTS). Please refer to [9] for details on the untimely delivery of messages in RaSTA.

Furthermore, the interlocking's RaSTA module was initially unable to connect to more than one field element at the time, because the implementation relied on the field elements establishing connections, while EULYNX specification assigns the connection startup to the interlocking. We fixed both issued and submitted our patch to the original authors. The corresponding pull request [20] was accepted and is now part of the public repository and available to further researchers. These results show that our environment allows us to validate the functionality of protocol implementations in realistic scenarios.

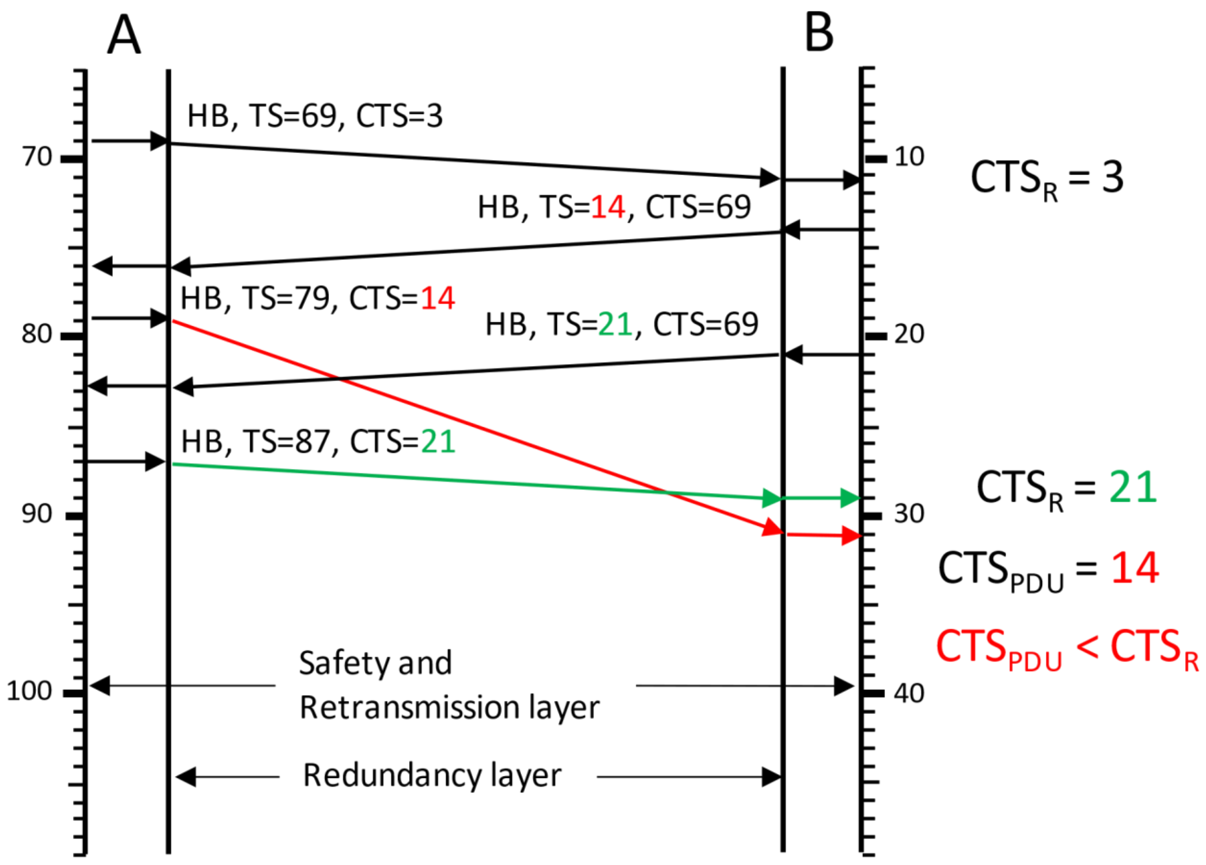

Figure 4: Observed behavior during debugging and the associated messages' relevant data fields. Heartbeat messages (HB) are exchanged, which include a timestamp (TS) and a confirmed timestamp (CTS). The last relevant CTS is stored in CTS_R, to be compared with newly arrived CTS in CTS_{PDU}.

Additionally, we instrumented our environment to measure the time spent in different communication steps, e.g., when moving a point machine to a new end position under different network conditions (see Figure 6). We confirm that the impact of increasing network latency is linear as expected, validating our environment's ability to perform quantitative assessments. Due to more apparent limitations with the used RaSTA implementation, we couldn't test the RaSTA protocols' behavior in networks under influence of packet loss since the loss of packets and subsequent arrival of unexpected packet sequence numbers lead to crashes of the RaSTA module. Addressing these was out of the scope of this paper.

The route setting time was averaged over 100 repetitions with a network parametrized to the measurements of the DB Netz AG and resulted in 26 ms. In our scenario, the point machine move duration is disregarded, meaning the point machine is assumed to instantly arrive at the new end position. Table 1 displays the mean and the standard error of the mean of each steps' duration. The distribution of durations of specifically network-dependent steps is shown in Figure 5.

Step	Mean [ms]	SEM [ms]	Actor	Process	Message
0	-	-	Interlocking	System	Move Point
1	0.637	0.021	RaSTA sends	System	Move Point
2	4.403	0.358	RaSTA receives	Network	Move Point
3	0.040	0.001	SCI-P receives	System	Move Point
3.1	0.555	0.011	SCI-P sends	System	Point Position
3.2	0.948	0.020	RaSTA sends	System	Point Position
3.3	5.445	0.288	RaSTA receives	Network	Point Position
3.4	0.101	0.004	Interlocking receives	System	Point Position
4	-	-	Point Machine moves/ SCI-P sends	System	Point Position
5	1.216	0.286	RaSTA sends	System	Point Position
6	5.669	0.325	RaSTA receives	Network	Point Position
7	0.117	0.002	Interlocking receives	System	Point Position
8	1.108	0.015	Interlocking sends	System	Indicate Signal Aspect
9	0.629	0.008	RaSTA sends	System	Indicate Signal Aspect
10	4.284	0.299	RaSTA receives	Network	Indicate Signal Aspect
11	0.048	0.001	SCI-LS receives	System	Indicate Signal Aspect
12	0.662	0.014	SCI-LS sends	System	Indicated Signal Aspect
13	0.931	0.012	RaSTA sends	System	Indicated Signal Aspect
14	6.155	0.298	RaSTA receives	Network	Indicated Signal Aspect
15	0.101	0.009	Interlocking receives	System	Indicated Signal Aspect

Table 1: Mean and standard error of the mean for individual durations of route setting steps

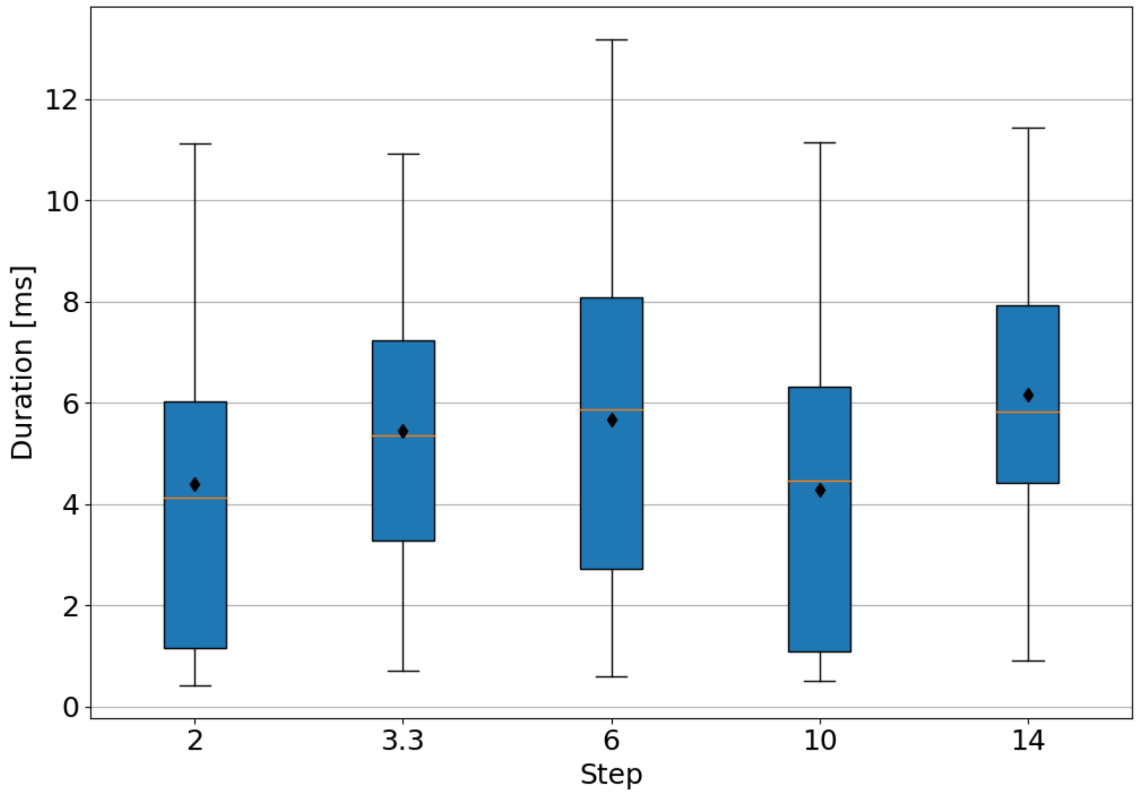

Figure 5: Distribution of durations of network-dependent steps during route setting

We also measured the bandwidth load on our environment's internal network using a tool provided by VirtualBox. The experiment entails the continuous exchange of messages for the sequential setting of two alternating routes and thus represents a worst-case scenario load-wise. During the procedure, we measured the bit rate going through both the blue and the grey switch of the interlocking to be 9.6 kbit/s for the SCI-P connection and the 7.47 kbit/s SCI-LS connection respectively. The difference is due to one additional message during route setting for SCI-P.

Although hardware-in-the-loop functionality is missing from the prototype, the resource monitor of VirtualBox allows for an evaluation of the required hardware capabilities to run the VM with respect to the host system. By extrapolation based on CPU benchmarks we estimate the OC module to comfortably run on a low powered single-board computer like the Raspberry Pi Zero 2 W.

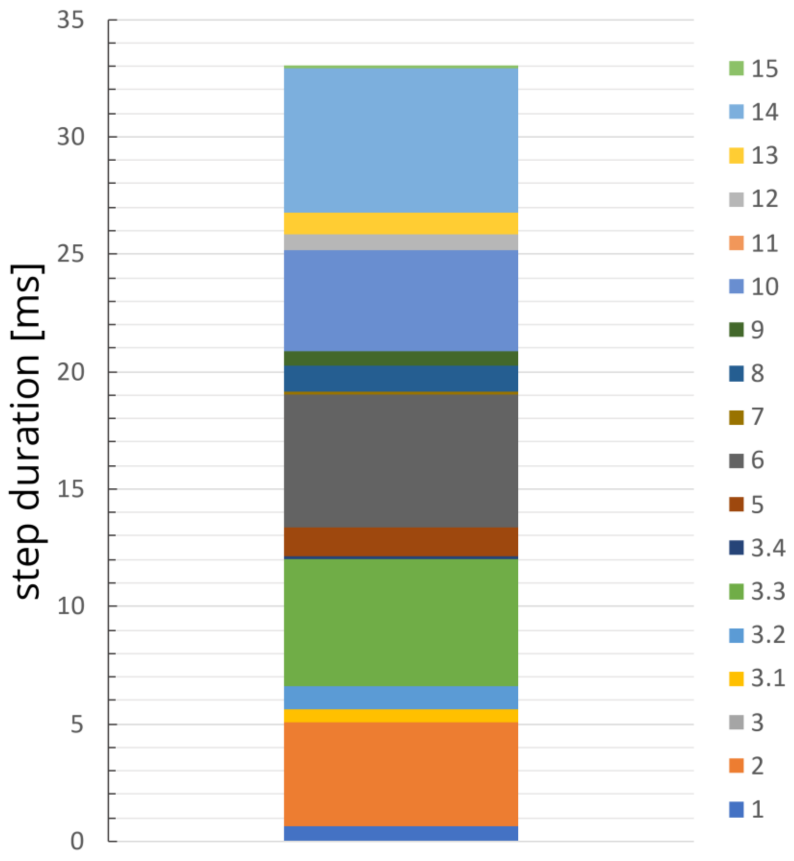

Figure 6: Aggregated route setting duration for all steps

5 Discussion

The test subject for the validation of our evaluation environment, the open source RaSTA implementation [7] in the version published 18th of July 2022 included some breaking bugs. After our changes described above, it managed to perform according to expectations in the limits set for the network latency by EULYNX of 50 ms. A possible explanation for the discovered errors in the used RaSTA implementation is that the authors only used environments based on local, container-based virtualization for their integration tests. Environments like the one proposed by us achieve a higher degree of realism and can help to reveal further flaws in IP based command and control systems.

The measured route setting time of 26 ms is multiple magnitudes lower than other figures found in literature for DSTW route setting times. Within the scope of the research conducted for implementation of ETCS for the new suburban railway network in and around Stuttgart, suppliers estimated route setting times of 7 to 9 s, disregarding point machine movement durations. This striking difference can only partially be explained by the simplifications made for our prototype. Properties of safe systems, whose omission might lead to a lower measurement of latency, include slower clocks in processors for safety-critical applications and the overhead required for M out of N voting algorithms. Still, the latencies provided by the suppliers at the time most likely relate to guarantees which they are ready to make and do not push the technologies limits. It is up to the infrastructure managers to request and reward the exploitation of the remaining optimization potential.

The experiment setup also allows for an estimation of the maximum amount of field elements connectable to a TSO at a time via one access network in terms of bandwidth. According to the information provided by DB Netz, the bbIP network will be capable of full-duplex, gigabit speed [4]. In the worst-case scenario of continuously setting new routes, one TSO could serve upwards of 104,000 field elements, excluding cryptographic overhead. Considering the planned outfit of OC with both crypto-box and VPN components, we estimate the traffic to increase by 596% in the worst-case scenario, resulting in 17,477 served field elements. However, we regard the usage of both crypt-box and VPN to be a questionable decision, since RaSTA based on TLS over TCP is already seen as sufficiently secure by the EULYNX specification. In terms of the number of required TSO, we estimate that the approximately 250,000 installed field devices in Germany could be served from between 3 and 15 TSO locations, depending on the scenario. However, additional redundancy might be desirable.

6 Conclusion and Future Work

This work presents a first step towards a fully virtualized evaluation of components and protocols for digital interlocking. Instead of creating a physical replica of the entire network, we create a virtualized replica that allows for simple reconfiguration. We showed that our environment could validate basic OC functionality and perform quantitative assessments.

However, further steps are necessary to perform unit and integration testing inside such an environment. For example, currently, we can only evaluate virtualizable components. Thus, the evaluation of, e.g., object controllers in combination with the controlled field elements is necessary for integration testing. Therefore, we plan to implement the hardware-in-a-loop capability that was accounted for in the infrastructure design to integrate real hardware inside our virtualized system. Furthermore, for now, the interlocking configuration is done manually. We intend to automatically generate the entire digital interlocking configuration files based on EULYNX or PlanPro configuration files.

For now, we use Vagrant on a single machine to provision our environment. Thus, this single machine limits the scale of our environment (for now, the environment is mainly limited by memory capacity). We plan to migrate our environment to a private cloud environment and add functionality to run in public clouds provided by, e.g. Amazon, Google or Microsoft to further simplify development.

The application and impact of the route setting times measured within the scope of our prototypal implementation is limited due to its simplifications, most importantly the missing safety features. An evaluation environment with fewer or no simplifications with respect to the actual interlocking, running on SIL4-certifiable hardware, can demonstrate the practical limits of processing speed for safe rail operation. Which we estimate will not deviate meaningfully from the results achieved in this paper.

7 Bibliography

[1] Marc Behrens, Mirko Caspar, Andreas Distler, Nikolaus Fries, Sascha Hardel, Jan Kressner, Ka-Yan Lau, and R Pensold. 2021. Schnelle Leit- und Sicherungstechnik für mehr Fahrwegkapazität. *EI-Der Eisenbahningenieur* (June 2021), 50–55.

[2] Jossekin Beilharz, Philipp Wiesner, Arne Boockmeyer, Lukas Pirl, Dirk Friedenberger, Florian Brokhausen, Ilja Behnke, Andreas Polze, and Lauritz Thamsen. 2021. Continuously Testing Distributed IoT Systems: An Overview of the State of the Art. Retrieved from http://arxiv.org/pdf/2112.09580v1

[3] Dr. Bernd Elsweiler, Nikolaus Fries, Christian Summen, and EBA Herausgeber. 2021. EBA-Infotage: Einordnung und Übersicht über das Programm Digitale Schiene Deutschland (DSD).

[4] Petros Matios. 2022. Persönliche Kommunikation.

[5] Mirko Caspar, Hardi Hungar, and Daniel Schwencke. Effizientes Testen modularisierter und standardisierter Stellwerkskomponenten. *SIGNAL+DRAHT* 110, 9/2018 .

[6] Silvia Pascual and Frits Makkinga. Reduzierung der Betriebskosten für ERTMS Level 2 durch Implementierung einer standardisierten Stellwerksschnittstelle. *SIGNAL+DRAHT (109) 9/2017*, 8.

[7] Railway-Ccs. 2023. rasta-protocol. *GitHub*. Retrieved from https://github.com/Railway-CCS/rasta-protocol

[8] Robert Schmid, Arne Boockmeyer, Lukas Pirl, Randolf Berglehner, Ibtihel Cherif, Andreas Korff, Bernd Elsweiler, and Andreas Polze. 2021. EULYNX-Live: Eine Methodik zum Validieren von Systemspezifikationen in hybriden Feldtests. *Signal+Draht* 6, 113 (2021), 24–31.

[9] 2015. *DIN VDE V 0831-200: Elektrische Bahn-Signalanlagen – Teil 200: Sicheres Übertragungsprotokoll RaSTA nach DIN EN 50159 (VDE 0831-159)*. Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE.

[10] 2017. Report on Industry Workshop held 25th January 2017. Retrieved from https://eulynx.eu/index.php/documents/presentations-given/39-20170125-workshop-industry-report/file

[11] 2017. NetEm - Network Emulator at Linux.org. Retrieved from https://www.linux.org/docs/man8/tc-netem.html

[12] 2019. The Linux PTP Project. Retrieved from https://linuxptp.sourceforge.net

[13] 2021. The Official YAML Web Site. Retrieved from https://yaml.org

[14] 2023. Vagrant by HashiCorp. *Vagrant by HashiCorp*. Retrieved from https://www.vagrantup.com

[15] 2023. EULYNX documents. Retrieved from https://eulynx.eu/index.php/documents/published-documents

[16] 2023. Oracle VM VirtualBox. Retrieved from https://www.virtualbox.org

[17] 2023. Docker: Accelerated, Containerized Application Development. *Docker*. Retrieved from https://www.docker.com

[18] 2023. The official home of the Python Programming Language. *Python*. Retrieved from https://www.python.org

[19] 2023. Rust Programming Language. Retrieved from https://www.rust-lang.org

[20] FixCTS_R bug SgtChrome/rasta-protocol@24ce71d. Retrieved November 24, 2022 from https://github.com/SgtChrome/rasta-protocol/commit/24ce71d5a24a32bb906a04b7a8101b04b5fa236c