RailRED: a Node-RED-based framework for modeling Train Control Management Systems

Alessandra Rizzardi*, Raffaele Della Corte†, Jesús F. Cevallos M.*, Vittorio Orbinato†, Simona De Vivo†, Sabrina Sicari*, Domenico Cotroneo†, Alberto Coen-Porisini*

*Dipartimento di Scienze Teoriche e Applicate - Università degli Studi dell'Insubria
via O. Rossi 9, Varese (VA), 21100, Italy

†Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione - Università degli Studi di Napoli Federico II
via Claudio 21, Naples (NA), 80125, Italy

{alessandra.rizzardi, jf.cevallosmoreno, sabrina.sicari, alberto.coenporisini}@uninsubria.it
{raffaele.dellacorte2, vittorio.orbinato, simona.devivo, domenico.cotroneo}@unina.it

***Abstract*—The modeling and simulation of Internet of Things (IoT) and Industrial IoT (IIoT) systems allow practitioners to obtain valuable insights into the system's behavior before their actual deployment in the field. Early designing permits the analysis of the interactions among the involved entities, evaluating the effects of modifications, and understanding the impact of failures on the system. In particular, this is exacerbated in the context of IoT/IIoT, which is characterized by multiple and heterogeneous subsystems, different processing levels, and communication protocols. In such a direction, recent innovations in IT devices have enabled the rail industry to gather information from Train Control and Monitoring Systems (TCMS) to check conditions constantly and prevent issues, thus improving reliability and safety and, in some cases, leading to cost-saving by optimizing maintenance resources. In such a scenario, this paper presents RailRED, a framework for simulating and prototyping a TCMS based on the Node-RED tool. In RailRED, the main TCMS subsystems are modeled using Node-RED flows, while the subsystem interconnections are performed through a low footprint and encrypted gateway based on the MQTT protocol. The proposal can also generate diagnostic data that mimic the behavior of a real-world TCMS. RailRED communication latency and its ability to generate diagnostic data have been analyzed, with the latter evaluated by using clusters of diagnostic events collected from a real-world TCMS running on a high-speed train.**

***Index Terms*—Railway Monitoring and Control Systems, Internet of Things Wireless Networks, System Prototypes**

I. Introduction

Modeling the behavior of a system is a valuable means to obtain a system representation that replicates its aspects of interest in the digital space. This is especially true in the context of Industrial Internet of Things (IIoT) systems, which are characterized by multiple and heterogeneous hardware/software components, different processing levels (e.g., the well-known edge-fog-cloud architecture), and communication protocols. An example is the Train Control and Monitoring System (TCMS) [1], which represents a train's brain and communication backbone, integrating and managing all onboard information through a wide set of heterogeneous subsystems. In this context, the use of modeling can enable the analysis of the system before its setup in an industrial environment, e.g., to accelerate the testing phase and reveal potential issues at an early stage, to understand the failure behavior of the system, evaluate the use of machine learning-based solutions for predictive maintenance or monitoring, etc., obtaining valuable insights on the target system.

Node-RED[1] represents one of the leading frameworks for modeling and simulating Internet of Things (IoT) systems. It is a flow-based development tool for visual programming, designed to connect hardware devices, APIs, and online services. Based on Node.js, Node-RED has a small footprint, allowing its deployment on embedded systems. Given its ease of use, flexibility, and ability to rapidly prototype and deploy complex applications, Node-RED has been used to model/simulate different types of IoT systems, such as indoor and outdoor scenarios [2], edge computing orchestration [3], and remote patients and building monitoring [4], [5].

This paper presents RailRED[2], a Node-RED-based framework for simulating and prototyping a TCMS, which has been the target of both industrial and academic efforts that have recently been made to deepen the automation and evolution of railway control and automation systems [1]. The framework models the main components characterizing a TCMS, such as Pneumatic Braking System (PBS) and Heating, Ventilation, and Air Conditioning (HVAC), and their intercommunication. RailRED integrates an easy-to-use set of Node-RED flows that simulates most of the TCMS subsystems. Subsystem flows are interconnected through a low-footprint and encrypted gateway leveraging the Message Queuing Telemetry Transport (MQTT) protocol. RailRED is also provided with a diagnostic data generator, which is able to mimic the behavior of a real-world TCMS by using clusters of events generated from real diagnostic data. RailRED is also equipped with a graphical user interface that enables direct monitoring and control.

The proposal has been designed in the context of the SERENA-IIoT project[3], which aims at enhancing knowledge in the next-generation IIoT systems by introducing novel solutions concerning reliability monitoring and security. In this

[1]https://nodered.org

[2]RailRED is publicly available at https://github.com/DIETI-DISTA-IoT/RailRED.

[3]http://serenaiiot.dista.uninsubria.it/

979-8-3503-8744-5/24/$31.00 ©2024 IEEE

context, we analyze the communication latency of RailRED, as well as its ability to generate diagnostic data. We used clusters of diagnostic data generated from a real-world TCMS running on a high-speed train for the latter.

II. Related Work

Using models and/or simulated systems, practitioners can easily analyze the system behavior, evaluate the effects of modifications (e.g., the update of existing components or the integration of new ones), as well as understand the impact of failures on the system. Different solutions exist for modeling IoT/IIoT edge systems. An example is NS3[4], which is a discrete-event network simulator supporting Wireless Sensor Networks (WSN) and Internet of Things network topology. CupCarbon[5] is a simulation tool mainly targeting Smart City and IoT-based WSN, which allows the design, visualization, debugging, and validation of distributed algorithms for monitoring, environmental data collection, etc., and the creation of environmental scenarios. Cisco packet tracer[6] is a simulation tool that can simulate smart devices, sensors, and drives.

One of the most used tools for modeling and simulating IoT/IIoT systems is Node-RED, an open-source flow-based programming tool that allows the development of IoT systems, enabling the integration of APIs, hardware devices, and web services. A Node-RED flow represents a system or a portion of a system, and it is characterized by a set of interconnected nodes, each one representing a system functionality. Nodes exchange messages in order to collaborate to complete the flow. Node-RED is built on Node.js, which requires few resources during execution, making it helpful to run at edge level on resource-constrained nodes [3]. In addition, Node.js takes full advantage of event-driven architectures, which makes Node-RED useful for IoT/IIoT systems. Considering such features, we selected Node-RED as the underlying tool of RailRED, in order to model/simulate a TCMS. Node-RED has been used to model different types of systems, such as indoor and outdoor monitoring applications [2]. The proposal is based on Node-RED, and leverages MQTT to interconnect the different system components. A workflow manager for the orchestration of services at the edge level is presented in [3]. The approach makes use of Node-RED, and encompasses a set of flows and REST endpoints for managing the upload of recipes, their instantiation as well as their execution as business processes. The work in [4] presents a patient monitoring system based on Node-RED, where data collected from the patient are collected and analyzed through a dedicated Node-RED flow, and made available to remote monitoring devices. Finally, in [5] Node-RED has been used in three different use cases, i.e., a smart home system for elderly monitoring interconnecting PLCs from different brands and a mobile robot, a smart door system with face recognition capabilities, and an automated system to monitor several buildings (each one equipped with sensors and actuators).

[4]www.nsnam.org

[5]www.cupcarbon.com

[6]www.netacad.com/courses/packet-tracer

With respect to other studies that exploit Node-RED for modeling and prototyping indoor and outdoor monitoring applications [2], this work leverages Node-RED to model a TCMS system to simulate the different components in order to easily enable the study and analysis of the system itself. The proposed framework is also able to generate diagnostic data mimicking the behavior of a real-world TCMS by using clusters of events obtained from actual diagnosing data.

III. Proposed framework

This section presents an overview of the main building blocks of RailRED, as shown in Fig. 2. All the components hereby explained are delivered inside a custom container-based environment to facilitate portability and avoid complex system configuration.

A. TCMS components

VCU and HMI. The Vehicle Control Unit (VCU) manages some of the main train propulsion capabilities, like traction and breaking. In RailRED, the VCU receives synthetic data from simulated sensors and issues automatic control signals, while manual commands such as auto-coupling and pneumatic braking are triggered from the Human-Machine Interface (HMI) subsystem. Automatic and manual commands are processed by the other subsystems modeled in RailRED, which simulate the Heating, Ventilation, and Air Conditioning (HVAC), the Passenger Information System (PIS), and the Pneumatic Braking System (PBS), among other functionalities.

HVAC. This subsystem includes temperature and humidity-simulated sensors, which are implemented as emitter nodes that generate random values sampling from a user-defined normal distribution. Based on the sensor inputs, reactive triggering of heating, cooling, ventilation, and de-humidifying signals are also provided in the HVAC. A portion of the HVAC flows is shown in Fig. 1.

PIS. This subsystem provides train passengers with information about the journey. In RailRED, these data are managed through a PostgreSQL database that contains station sequences, estimated arrival times, and distances. Arrival times are estimated with fixed parametric transformations of the distances.

PBS. The subsystem manages the train's pneumatic brakes. In RailRED, they can be gradually activated from the HMI. *Spline* linear functions simulate the effects of manual braking on the train speed, brake temperature, and air pressure. Electric braking and acceleration are instead simulated using the identity function with respect to the control signal.

Other functionalities. The HVAC simulated sensoring and reactive triggering mechanisms are replicated for compressed air production, door control, and wi-fi service control.

Component interconnection. A symmetric-encrypted gateway interconnects all the modeled TCMS subsystems and is built over MQTT, Mosquitto[7], and Rabbit [6]. Sensory inputs and commands are transferred between components

[7]https://mosquitto.org

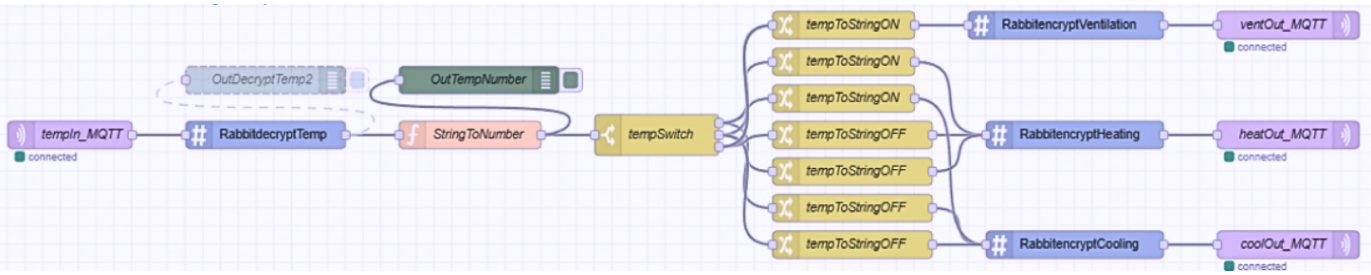

Fig. 1. A schematic diagram of the components of the reactive environment control module.

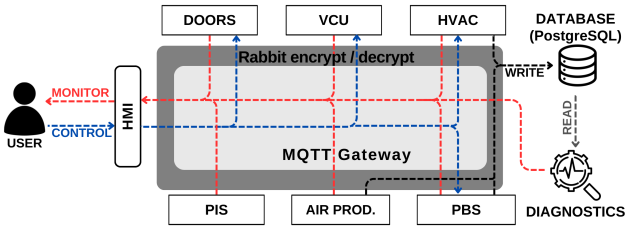

Fig. 2. RailRED architecture overview.

as MQTT messages, which are organized in a hierarchical topic taxonomy. Messages are encrypted using the Rabbit cipher with topic-specific encryption keys. In addition to the symmetric encryption of the payload, the SSL/TLS client-server security scheme is used at the transport level.

B. Diagnostics

Diagnostic events are the primary information source for understanding a system's runtime behavior. They are usually characterized by sequences of text lines reporting on the runtime behavior of a system [7], including entries highlighting the occurrence of errors/failures. Their analysis has been extensively used for troubleshooting [8]–[10].

To emulate the generation of diagnostic events related to the TCMS runtime, RailRED has been equipped with a diagnostic subsystem. This subsystem aims to generate realistic diagnostic data starting from a database (DB) of events observed in real scenarios. The database encompasses clusters of events, i.e., events generated by the multiple subsystems composing a TCMS, which are strictly related and close in time. In this study, the database encompasses clusters of events obtained by analyzing one week of data (accounting for more than 57k diagnostic events) generated by a real TCMS system of a high-speed train during operation. The analysis of raw data encompasses two main steps, which are performed beforehand: (i) tupling, i.e., a recognized approach for temporal coalescence [11], with the aim to group the events that are close in time (generating a tuple), since they can be related to the same underlying anomaly, and (ii) hierarchical clustering [12], with the aim to create clusters of tuples encompassing similar sets of events. At the end of the analysis, the clusters DB encompasses 249 clusters, with each one characterized by a set of events (i.e., the events composing the tuples used to generate the cluster) and its average duration.

Given the clusters DB, the diagnostic subsystem loads the clusters at startup and generates a temporal profile to use for diagnostic events generation, i.e., a set of timestamps indicating when diagnostic events should be generated. To this aim, the diagnostic subsystem makes use of the Weibull reliability model, i.e., $R(t)=e^{-(\lambda \cdot t)^{\alpha}}$, where λ is the *scale* parameter and α is the *shape* parameter (in this study we used $\lambda = 0.000001$ and $\alpha = 0.92$). The *Weibull* is among the most used distributions in failure analysis [13], [14], although any other model would have fit the aim of our emulation. When one of the timestamps of the temporal profile occurs, the diagnostic subsystem:

1) Randomly selects one of the loaded clusters;
2) For each event composing the cluster: (i) Generates the event with the current timestamp. It should be noted that the events composing the cluster are generated considering the average duration of the cluster; therefore, some delay is introduced between the events to emulate the real scenario described by the cluster. (ii) Writes the event in a diagnostic table of the gateway database.
3) Waits for the next timestamp of the temporal profile.

TABLE I
DIAGNOSTIC EVENTS EXAMPLE.

Timestamp	*Event code*	*Event description*	*Component*
2024-06-12 23:42:45	9921	Lost communication with HVAC in cabin 4	HVAC_4
2024-06-12 23:42:47	1021	Pressure sensor 16 fault in cabin 4	HVAC_4

TABLE I depicts some examples of events generated by the diagnostic subsystem. These events are made available to the train operators through a dedicated dashboard, where the main fields of the events are shown, i.e., *Timestamp*, *Event code* (identifying the type of event), *Event description*, and *Component* (providing the component the event refers to).

IV. Latency and Diagnostic Analysis

We analyze the TCMS's performance in terms of communication latency between data generation and data visualization nodes. More specifically, the measurements regarded the latency between temperature and humidity probe generation at the HVAC and its corresponding visualization at the HMI dashboard. Six different experiments are made using three different probing frequencies: 1 Hz, 0.2 Hz, and 0.1 Hz, respectively. For each frequency, two experiments are made, collecting latency measurements for 30 minutes (*Short*) and 60 minutes (*Long*), respectively. Box plots in Fig. 3 summarize the obtained results. Note that message latency goes rarely

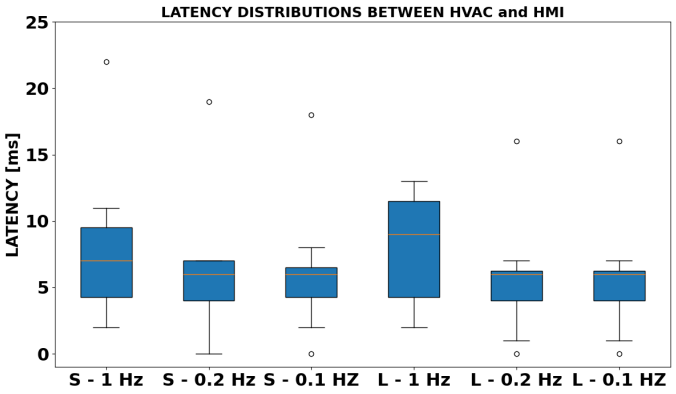

Fig. 3. Commnication latency distributions for different batch runs.

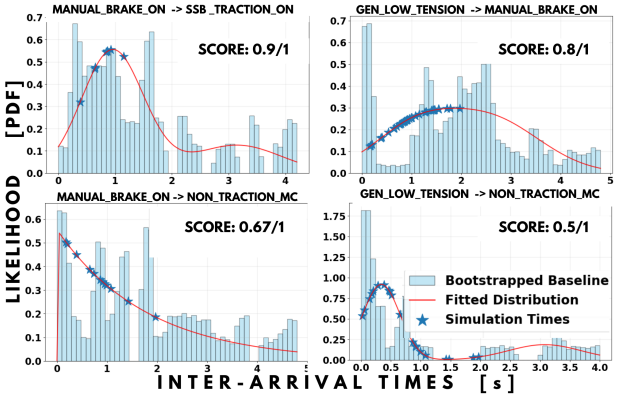

Fig. 4. Empiric validation of intra-cluster inter-arrival times.

beyond 10 ms, which suggests RailRED to be a promising TCMS codebase for more compound projects to be built upon.

Furthermore, the realism of our diagnostic event generator is validated in terms of how the generated inter-arrival times between events in the same cluster resemble the real distributions observed in the real DB, mentioned in Section III-B. As shown in Fig. 4, the inter-arrival times in the real traces are extended using the bootstrap method [15]. Then, exponential or multimodal Gaussian distributions are fitted to such data and considered to obtain the prior distributions for event-specific inter-arrival times. Finally, the likelihood of the generated inter-arrival times under such priors is computed and scaled with respect to the maximum value in each probability density function (red lines). The plot-wise scores in Fig. 4 correspond to the mean value of such scaled likelihoods (blue stars). It can be noted that the generated inter-arrival times (blue stars) in RailRED generally obey observations from real traces (light blue histogram and red fitted probability density functions).

V. Conclusions

This paper presents RailRED, a framework for TCMS modeling and simulation based on Node-RED. RailRED leverages encryption-aware, interconnected flows that simulate common train subsystems and a user-friendly control and monitoring dashboard. It is also able to generate realistic diagnostic data based on real system events. We analyzed the communication latency of our solution as well as its ability to generate realistic diagnostic data, obtaining some promising preliminary results. Future versions of this framework will improve existing flows through signal feedback cycles and the implementation of machine-learning-based event generation for a more fine-grain emulation of real diagnostic event occurrence pace.

Acknowledgments

This work was supported in part by the SERENA-IIoT project, which has been funded by MUR (*Ministero dell'Università e della Ricerca*) under the PRIN 2022 program (project code 2022CN4EBH), and by the project SERICS (project code PE00000014), under the NRRP MUR program funded by the EU - NGEU. MUR has also partially supported this work for research activities - PON Research and Innovation 2014-2020; Action IV.5 "Doctorates on green topics" - CDA Resolution No. 73 of 29.04.2022. UGOV: 000010—PON_DOTT_GREEN_ITEE_37_CICLO_001_005, and by the GENIO project (CUP B69J23005770005).

References

[1] J. Goikoetxea, D. O. de Eribe, A. Grasso, and B. Bienfait, "Shift2Rail TAURO: Technologies for the Autonomous Rail Operation," *Transportation Research Procedia*, vol. 72, pp. 3719–3722, 2023.

[2] A. Rizzardi, S. Sicari, and A. Coen-Porisini, "Towards rapid modeling and prototyping of indoor and outdoor monitoring applications," *Sustainable Computing: Informatics and Systems*, vol. 41, p. 100951, 2024.

[3] F. Larrinaga, W. Ochoa, A. Perez, J. Cuenca, J. Legaristi, and M. Illarramendi, "Node-red workflow manager for edge service orchestration," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–6.

[4] S. Sicari, A. Rizzardi, and A. Coen-Porisini, "Home quarantine patient monitoring in the era of COVID-19 disease," *Smart Health*, vol. 23, p. 100222, 2022.

[5] H. Wicaksono, P. Santoso, I. Putro, I. Hutomo, and P. Alvina, "Towards integration of heterogeneous controllers in an IoT-based automation system," *E3s Web of Conferences*, vol. 188, p. 00009, 2020.

[6] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: A new high-performance stream cipher," in *10th International Workshop on Fast Software Encryption*. Springer, 2003, pp. 307–329.

[7] M. Cinque, R. Della Corte, and A. Pecchia, "An empirical analysis of error propagation in critical software systems," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2450–2484, 2020.

[8] J. Tian, S. Rudraraju, and Z. Li, "Evaluating web software reliability based on workload and failure data extracted from server logs," *IEEE Transactions on Software Engineering*, vol. 30, no. 11, pp. 754–769, 2004.

[9] E. Chuah, A. Jhumka, J. C. Browne, B. Barth, and S. Narasimhamurthy, "Insights into the diagnosis of system failures from cluster message logs," in *11th European Dependable Computing Conference (EDCC)*, 2015, pp. 225–232.

[10] A. Pecchia, I. Weber, M. Cinque, and Y. Ma, "Discovering process models for the analysis of application failures under uncertainty of event logs," *Knowledge-Based Systems*, vol. 189, p. 105054, 2020.

[11] C. Di Martino, M. Cinque, and D. Cotroneo, "Assessing time coalescence techniques for the analysis of supercomputer logs," in *IInternational Conference on Dependable Systems and Networks*. IEEE, 2012, pp. 1–12.

[12] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[13] T.-T. Lin and D. Siewiorek, "Error log analysis: statistical modeling and heuristic trend analysis," *IEEE Trans. on Reliab.*, pp. 419–432, 1990.

[14] M. Cinque, D. Cotroneo, R. Della Corte, and A. Pecchia, "A framework for on-line timing error detection in software systems," *Future Generation Computer Systems*, vol. 90, pp. 521–538, 2019.

[15] B. Efron and B. Narasimhan, "The automatic construction of bootstrap confidence intervals," *Journal of Computational and Graphical Statistics*, vol. 29, no. 3, pp. 608–619, 2020.