Assessing the Performance and Impact of an Open-Source RTI in Distributed Real-Time Military Simulation

João Rodolfo de Oliveira Rosa[1], **Álvaro Luiz Fazenda**[1]

[1]Federal University of Sao Paulo - Science and Technology Institute
Cesare Monsueto Giulio Lattes Avenue 1201, Sao Jose dos Campos, Sao Paulo, Brazil

{rodolfo.rosa,alvaro.fazenda}@unifesp.br

Abstract. *This study evaluates the performance of PORTICO, an open-source Runtime Infrastructure (RTI), for real-time military simulations. The research compares its efficiency with the commercial pRTI solution, analyzing key high-level architecture (HLA) services, including federate interactions and attribute updates. The results demonstrate the viability of the open source solution even beyond conventional usage scenarios, accounting for federation configuration parameters and composition. Furthermore, the work establishes foundations for developing a Minimum Viable Product (MVP) for Modeling and Simulation as a Service (MSaaS) using HLA concepts, enabling scalable and efficient distributed simulations.*

Resumo. *Este estudo avalia o desempenho do PORTICO, uma Infraestrutura de Tempo Real (RTI) de código aberto, para simulações militares em tempo real. A pesquisa compara sua eficiência com a versão comercial pRTI, analisando serviços-chave da Arquitetura de Alto Nível (HLA) como interações entre federados e atualizações de atributos. Os resultados indicam que a solução de código aberto é viável mesmo em cenários além do uso convencional, considerando parâmetros de configuração e composição da federação. Adicionalmente, o trabalho estabelece bases para desenvolver um Produto Minimamente Viável de Modelagem e Simulação como Serviço utilizando conceitos HLA, possibilitando simulações distribuídas escaláveis e eficientes.*

1. Introduction

This article explores the feasibility of using PORTICO software as a middleware to integrate simulators in real-time. PORTICO is an RTI (Run-Time Infrastructure), the core component of HLA (High Level Architecture). RTI is a product that has been consolidated on the market with commercial and open-source implementations. HLA is the market-adopted standard for integrating simulators in the military context. Maintained by the Simulation Interoperability Standards Organization and documented in specific IEEE (Institute of Electrical and Electronics Engineers) standards, the key ones include 1516-2010, 1516.1-2010, and 1516.2-2010 ([IEEE 2010b], [IEEE 2010a],[IEEE 2010c]).

The article seeks to evaluate the robustness, performance, and productivity of this open-source tool, which incurs no usage license costs and technology domain, in comparison to a commercial solution. The objective is to evaluate the potential advantages and drawbacks of opting for an open-source tool over established commercial software for simulator integration. In analyzing the advantages of open source software for the

defense sector, [Russo 2016] asserts that they offer high reliability attributed to the active development community, allowing a reduction in the costs of implementing new systems. The author further clarifies that the utilization of open source software prevents defense forces from becoming overly reliant on proprietary technologies, often associated with single suppliers, thus mitigating long-term costs.

The evaluation of the capabilities of PORTICO includes assessing its performance concerning the ability to scale the number of federates, identifying any limitations that may arise when increasing the size of messages shared between federates, and exploring the system's behavior when different callback models are employed. Productivity analysis examines both RTI and the experiments that extend the tests established by [Gütlein 2020]. This investigation contributes to the field by evaluating middleware performance for distributed systems with real-time requirements, distributed systems performance, and foundational software for simulator integration.

Although RTI has been widely studied, few works have assessed real-time performance or usability. Existing research prioritizes throughput/scalability over deployment practicality. This study addresses two gaps: (1) real-time evaluation of open-source RTIs (PORTICO) and (2) HLA-based MSaaS proposals using open-source tools. Key contributions include: PORTICO's empirical testing against Nielsen's 0.10s threshold; pRTI comparison under matched conditions; usability analysis (installation/docs/support); and an open-standards MSaaS architecture.

2. Background

HLA encapsulates essential principles related to distributed systems and distributed simulation. In this section, the aim is to introduce these fundamental concepts, providing a comprehensive understanding. In addition, other specialized concepts unique to HLA and its associated research endeavors are explored.

2.1. Simulation in the military context

Within the sphere of military activities, simulation has received considerable importance. Specifically, the domain where HLA originated and where it finds extensive application is examined. Simulation is vital in military contexts, offering risk reduction, cost-effectiveness, and improved combat readiness. According to [Kuhl et al. 1999], the end of the Cold War in the 1980s reshaped military planning and resource allocation. Advances in computer systems helped address shrinking defense budgets, making computational simulation a powerful tool for tackling complex scenarios cost-effectively. As noted in [Kuhl et al. 1999], simulations allow the evaluation of new organizational structures, techniques, and doctrines using existing technology. They also allow for checking the effectiveness of new weapons systems before acquisition. In summary, simulation supports informed decision making, optimizes training, and ensures combat readiness.

2.2. Distributed real-time simulation

Military simulations, with their intricate interplay of diverse entities and scenarios, defy the feasibility of monolithic systems. Consequently, a paradigm shift toward distributed simulations becomes essential. In this context, the simulation environment should seamlessly present users with a unified view, a concept similar to that of distributed systems.

According to [Tanenbaum and Steen 2007], a distributed system consists of independent computers working as a unified entity. HLA aims to integrate simulators, promote interoperability, and enable seamless distributed simulation. Transitioning from monolithic to distributed systems enhances military simulations' effectiveness and realism, supporting informed decision-making and operational readiness. In turn, [Erciyes 2019] characterizes real-time systems based on the timeliness of responses to requests and the associated deadlines. According to the author, real-time systems operate with temporal constraints, imbuing significance into the deadline for responding to requests.

Using the definition of distributed systems as a collection of independent computers that present themselves as a single system, the concept of distributed systems in real time emerges. Such systems constitute a collection of independent simulation systems interconnected by a real-time network with a well-defined response time for transactions. This article adopts the response time limit definitions proposed by [Nielsen 1994]: 0.10 s, 1.00 s, and 10.00 s. Users perceive responses within 0.10 seconds as instantaneous. In the range between 0.10 s and 1.00 s, although users lose the sense of direct interaction with application data, their attention remains uninterrupted. The 10.00 s limit, as defined by [Nielsen 1994], was not applied in this study. This decision is based on the fact that user interaction in the discussed context primarily involves dialogues, where more immediate responses, such as those within the range of 0.10 to 1.00 s, are more relevant.

2.3. HLA

The HLA standard was developed as a demand-driven solution by the US Department of Defense (US DoD) in the 1990s to integrate simulators. According to [Kuhl et al. 1999], the main objective of this integration was to enable simulator interoperability and facilitate the reuse of their components. Key concepts related to HLA, as outlined in [Moller 2012], include the following:

- RTI: It is the core component of the HLA, serving as a middleware that facilitates message exchange between integrated applications. It consists of two software elements: a central component (an application with a network address) and local components (libraries installed on federated systems). Federates interact with the RTI through their local component, typically by instantiating an RtiAmbassador object. The RTI, in turn, communicates with the federate via callbacks, requiring the use of a Federate Ambassador object instantiated within the federate. All interactions between the federate and the RTI occur through method calls, initiated by the federate (calls) or the RTI (callbacks), as defined by [IEEE 2010a]. This interaction framework is known as the callback model.
- Federate: Simulators are typically the applications to be integrated, but can also include physical devices connected via a data network. As noted in [Kuhl et al. 1999], a federation can act as a gateway for human participants to join, serve as data collectors, or function as data viewers. Additionally, physical components (hardware) can be integrated into virtual environments for testing purposes.
- FOM (Federation Object Model): a document that standardizes the data shared between federates. The shared data are defined as object classes (objectClass) and interaction classes (interactionClass). The FOM is specified in the object model template ([IEEE 2010c]);

- Federation: the set formed by the RTI, the federates, and the FOM. It represents the integrated environment where simulators and other applications interact, sharing standardized data based on the FOM.

In summary, HLA provides a framework for interoperability among simulators, ensuring standardized communication and the exchange of data in a cohesive and reusable manner within a federated environment.

2.4. RTI implementations - PORTICO and pRTI

PORTICO is an open source RTI that allows the implementation of federations within the HLA framework, as described by [Pokorny and Fraser 2023]. Developed in Java, it lacks a graphical interface and uses the JGROUPS library for federated communication. The project began in 2005, led by [Pokorny and Fraser 2023], and received funding from the Australian Defense Simulation Office in 2007, as noted by [Tu et al. 2012]. To understand PORTICO's implementation, a study of the JGROUPS library was conducted, involving downloading the library, running demonstration programs, and reviewing its documentation. Notably, aspects of JGROUPS' implementation, particularly its indirect communication paradigm in distributed systems, align closely with HLA services. Additionally, PORTICO includes an RTI Initialization Data (RID) file, which allows the configuration of parameters such as data packet size and network routing.

In contrast to PORTICO's decentralized approach, the RTI developed by the Swedish company Pitch Technologies employs a centralized structure, where a central RTI component manages message transmission between federates. Unlike PORTICO, it includes a graphical interface, improving usability and productivity, and does not require a RID configuration file. For testing and learning, Pitch Technologies provides a free version that supports up to two federates, allowing users to explore its features within a limited scope.

2.5. Related Work

The study conducted by [Gütlein 2020] involved a performance evaluation of four different RTI implementations: MÄK RTI, Pitch pRTI, PORTICO, and CERTI. The authors highlighted various advantages of a distributed simulation architecture, including the ability to overcome physical memory limitations, the ability to achieve performance gains, and the ability to improve fault tolerance.

Although [Malinga and Le Roux 2009] analyzed the performance of RTI, their study is over a decade old and is based on outdated middleware versions. Recognizing the ongoing complexity of HLA implementation, even as a mature technology, [Falcone and Garro 2018], [Falcone 2017a], and [Falcone 2017b] explored related research. They proposed frameworks such as reactive HLA (RxHLA) and Development Kit Framework (DKF) to simplify simulator integration.

Further research using PORTICO, such as [Amponsah 2019], [Grogan and de Weck 2018], and [Gütlein and Djanatliev 2020], has demonstrated the viability of RTI in production environments. For example, [Amponsah 2019] applied PORTICO to study the usage of electricity in large-scale urban buildings, highlighting the availability of documentation as a key factor. Similarly, [Gütlein and Djanatliev 2020]

used PORTICO for urban energy management, also emphasizing the importance of accessible documentation in their decision-making process.

The encouragement of the adoption of open source technology in the military domain is emphasized by [Russo 2016], who pointed out that it reduces long-term implementation and maintenance costs. The author also addresses the issue of technology independence in this context.

3. Experimental design

The experiments were carried out using 16 virtual machines managed via the Xen Orchestra administration tool and the XCP-NG hypervisor. The hardware included an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz and an Integrated Matrox G200eW3 Graphics Controller. Each virtual machine ran Debian 11 (bullseye) with 4GB of RAM.

To monitor memory usage in virtual machines, the Linux tool ”top” was utilized, which provided real-time information regarding memory resource consumption during the experiments. The experiments used pRTI version 5.4.5.0 (build 3195-Free), suitable for learning and testing purposes [Technologies 2023]. Furthermore, PORTICO version 2.1.0 was employed; its source code is available at [Pokorny 2023], while an executable version can be downloaded from [Pokorny and Fraser 2023].

The source code, adapted from the work of [Gütlein 2020], is available on GitHub, as referenced in [Oliveira Rosa 2023]. Refactoring was necessary, as the original work was in C++, and the adaptation ensured compatibility with the project's goals and requirements. The FOM defined by [Gütlein 2020] was used in the experiments. The experiments included interaction tests involving a shared vector of integers among federates and updates to attribute values. In the latter case, the integer attribute value of the object class was incremented by one unit.

Experiment 1 included both the interaction test and the update of the attribute value, while Experiments 2 and 3 focused solely on the interaction test, as detailed in the subsections on performance regarding message size and the pRTI vs. PORTICO comparison. Additionally, although not a formal experiment, participant feedback on installation and usability challenges for both RTI was documented.

Table 1 presents a summary of the experiments conducted. Experiment 1 used the EVOKED callback model and analyzed scalability by progressively increasing the number of federates, while exchanging fixed-size messages (an array consisting of four integers). Experiment 2, in turn, evaluated performance based on variations in message size and the bundling parameter, keeping the number of federates fixed at 16. Finally, Experiment 3 enabled a direct performance comparison between the open-source RTI, PORTICO, and the commercial RTI, pRTI, using both the EVOKED and IMMEDIATE callback models. For each experiment, depending on its objective, the total execution time or the average execution time was measured.

3.1. Experiment 1: Scalability

The objective of the experiment was to assess scalability by evaluating whether variations in processing time exhibited a linear relationship with the increase in the number of federates. Furthermore, the study aimed to determine whether the average execution time met the real-time application parameters outlined by [Nielsen 1994].

Table 1. Summary of Experiments

Experiment	Federates	Message Size	Callback Model	Iterations	Metric
Exp. 1	2-16	4 ints	Evoked	10.000	Average time
Exp. 2	16	10-80 KB	Evoked	2.500	Total time
Exp. 3	2	4 ints	Both	2.500	Average time

To achieve this, we initially deployed two virtual machines and gradually increased their number, doubling the count up to a maximum of 16 (2, 4, 8, and 16). The experiment consisted of 10,000 iterations that included interactions between the federates and updates of the attribute values. This process was repeated 10 times, measuring the total time for each set of 10,000 iterations in nanoseconds. The EVOKED callback model was used for communication, synchronizing both federate interactions and attribute value updates with time requests from the federate unit. The callback model concept was introduced in the subsection dedicated to the High-Level Architecture (HLA).

3.2. Experiment 2: Performance regarding message size

The study's manipulated variables included the size of messages transmitted by the federates and the time value set in portico.jgroups.bundling.maxTime, a parameter in the RTI configuration file associated with the JGROUPS library. This variable, part of JGROUPS Bundling Support, defines the maximum time for bundling messages before transmission. The messages were composed of arrays of integers with sizes ranging from 10 KB to 80 KB. The measured variable was the total execution time of 2,500 iterations. The federation consisted of 16 federates, and the selected callback model was EVOKED.

3.3. Experiment 3: pRTI and PORTICO comparison

The experiment aimed to compare the total and average times for an interaction test involving the exchange of a four-position integer array between two federates. A total of 10 series, each with 2,500 iterations, were conducted, and the total time for each series was recorded. Both callback models, EVOKED and IMMEDIATE, were tested, along with a comparative analysis of pRTI and PORTICO.

4. Experimental results

This section presents the results of the scalability experiments, the performance concerning the message size, and the comparison between pRTI and PORTICO. Each experiment includes figures depicting the time measurements obtained, accompanied by considerations regarding the results.

4.1. Experiment 1: Scalability

The data collected, which represents the elapsed times for the interaction test among the federates, involved transmitting an array of four integers, followed by a time advancement. Figure 1 presents these data as a box plot, illustrating the variability between the first and third quartiles (highlighted in yellow and green, respectively), along with the minimum and maximum values. The median is also indicated by a dividing line between the quartiles.

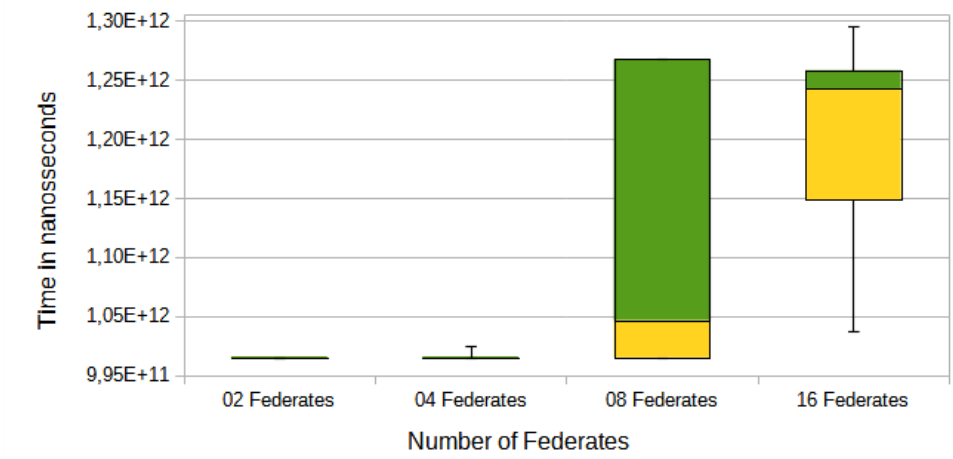

Figure 1. Scalability: interaction

Figure 1 shows that for 2 and 4 federates, the collected values have low variability and amplitude, with a median close to 0.10 s, as desired. However, with 8 federates, variability increases significantly, nearing the amplitude, although the median remains just above 0.10 s, close to the limit. Similarly, for 16 federates, variability decreases, but amplitude increases, with a median response time just above 0.12 s - still near the [Nielsen 1994] threshold, where users perceive the system to react instantly.

Furthermore, the graph shown in Figure 2 reveals that there was minimal variation in times for up to 4 federates. The data dispersion is negligible for 2 and 4 federates, leading to the Boxplot being nearly indistinguishable from a horizontal line. For two and four federates, the median is 1.01E+12 ns for 10,000 iterations, resulting in an approximate time of 0.10 s per iteration. This duration is sufficient for users to perceive the system as operating on the data directly and instantaneously.

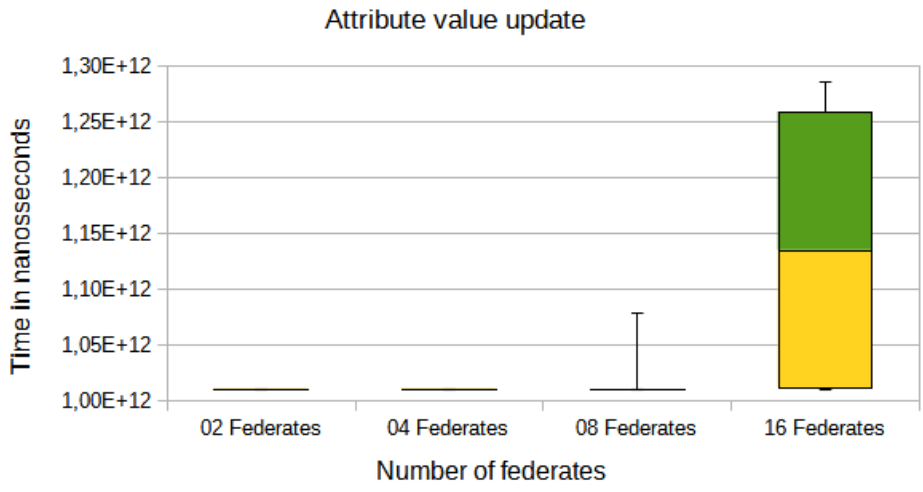

Figure 2. Scalability: attribute update

Upon examination of the same figure 2 for eight federates, a greater amplitude is

apparent compared to two and four federates, with a difference between the maximum and minimum values of 6.86E+10 ns. However, there is minimal variability in the data, with a median of 1.01E+12 ns for 10,000 iterations, indicating the adherence to the time limit defined by [Nielsen 1994]. The most notable variability was observed for 16 federates, at 2.47E+11 ns, differing by three orders of magnitude from the variability for 2, 4, and 8 federates. Examination of the median revealed a value of 1.13E+12 ns, which remained compatible with the response time limit values employed in this investigation.

4.2. Experiment 2: Performance regarding message size

As discussed in Section 2.2, Figure 3 illustrates the variation in total execution time regarding the message size shared between federates and the maximum time attribute of the JGROUPS bundle support. The portico.jgroups.bundling.maxTime parameter defaults to 30 milliseconds. However, Figure 4 shows irregular execution times for a 40 KB message, with a high standard deviation. This behavior stabilizes, with a lower standard deviation, only at 500 ms for JGROUPS Bundling Support.

Figure 3. Performance with 16 federates regarding Message Size for default parameter portico.jgroups.bundling.maxTime

The probability of message traffic, especially for larger messages of 40 or 80 KB, involving all federates sending and subscribing to such messages, is considered unlikely. It is important to note that the experiment deliberately imposed an extreme condition on the RTI to evaluate its behavior under challenging circumstances.

Despite the need for adjustments on the RTI parameters, the obtained results indicated times of less than 1.00 s. For message sizes up to 20 KB, the times were very close to the limit of 0.1 s with the standard configuration. Consequently, the RTI complies with real-time application requirements, as defined by [Nielsen 1994], albeit with certain limitations or restrictions. This analysis underscores the need to adjust the RTI execution parameters by editing the RID file. This requirement emphasizes that PORTICO demands comprehensive training, ensuring that users understand the various configuration possibilities to achieve optimal results.

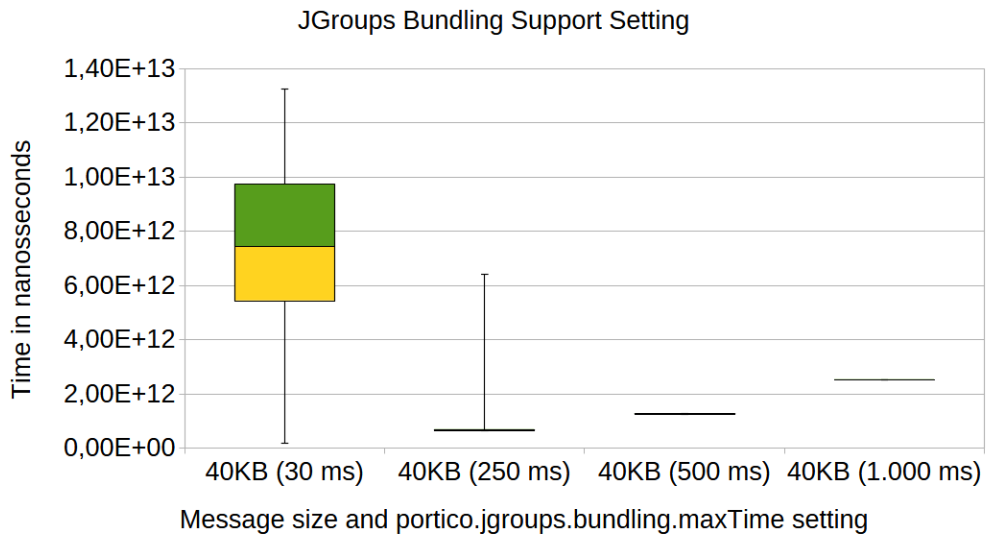

Figure 4. Performance with 16 federates for fixed Message Size (40KB) varying the parameter portico.jgroups.bundling.maxTime

4.3. Experiment 3: pRTI and PORTICO comparison

This experiment does not aim to determine the best RTI, but rather to assess its compliance with the required real-time parameter. The performance of both RTIs is shown in Table 2. The elapsed time for the pRTI immediate callback experiment is negligible considering two decimal places. Based on the values in Table 2 and the standard defined by [Nielsen 1994], both RTI meet the 0.10-second threshold, ensuring that the user perceives them as real-time systems.

Table 2. Time average comparison pRTI e PORTICO

RTI	Callback model	Iteration average time s
pRTI	Evoked	0.10
PORTICO	Evoked	0.10
pRTI	Immediate	0.00
PORTICO	Immediate	0.05

4.4. Productivity

Unfortunately, obtaining the license costs for pRTI proved challenging, despite its significance for comparison with PORTICO, particularly its open-source nature. To assess and assign value to the productivity aspect, we considered several criteria: ease of use, ease of installation, documentation, and support. Next, a summary of the key aspects examined.

1. Ease of use
   (a) pRTI: Graphical user interface that simplifies the interaction with the RTI and the setting of parameters.
   (b) PORTICO: Relies on a command-line interface and requires manual editing of execution parameters in text files.
   (c) Ease of installation

i. pRTI: Offers a streamlined setup—once the installation file (.exe or .deb) is downloaded, the process is straightforward and includes automatic configuration of environment variables.
ii. PORTICO: Requires additional steps, including the installation of a specific Java version and manual configuration of environment variables.

2. Documentation
   (a) In addition to the operation manual, a tutorial is provided, including demonstration software and practical examples to illustrate the primary HLA concepts. In addition, the company offers HLA courses in person and supports deeper understanding and proficiency.
   (b) PORTICO: The documentation is not regularly updated and presents a relatively shallow learning curve. Moreover, configuring parameters in the RID file lacks intuitiveness.
3. Support
   (a) pRTI: Upon contracting with the company, all the necessary support for installation and configuration is readily available, including priority support, software updates, license level adjustments, and even license transfers to other users, ensuring comprehensive support throughout the process.
   (b) PORTICO: The GitHub platform hosts a community maintained by RTI developers where various issues are discussed, also providing space for new questions to be solved by the community. However, there are no formal support services.
4. Costs
   (a) pRTI: It was not possible to determine the cost of the pRTI license. However, based on informal conversations, it seems that the cost may vary depending on the number of federates to be integrated. Information about the cost of licenses is not available on the company's website.
   (b) PORTICO: Distributed under the CDDL license (Common Distribution and Developer License), PORTICO ensures free use and distribution, permitting modifications to the source code. The absence of limitations related to the use of RTI allowed testing with 16 federates.

Furthermore, the use of open source software in defense applications, as highlighted in [Russo 2016], reduces costs and avoids the dependence on single vendors. However, integrating a simulator with RTI—a complex step that requires advanced knowledge for interface development—is facilitated in pRTI by the Pitch Developer Studio, while in PORTICO it relies on specialized programmers.

5. Conclusion

This research underscores the viability of employing an open-source RTI. Despite some limitations inherent to PORTICO, the primary operations in a distributed simulation were successfully executed, achieving a performance close to those defined as limits in this study for a real-time system. Verifying whether PORTICO implements all the services listed in the specification [IEEE 2010a] proved a challenge. Additionally, the lack of a graphical interface, requiring manual editing of the RID file for system configuration, may hinder usability compared to commercial RTI. However, access to the source code enabled

a detailed understanding of the implementation of the HLA service and provided insights into how early developers addressed project needs using the JGROUPS Java library.

It was not possible to evaluate the scalability limits of PORTICO compared to pRTI because the available test version was used. Considering that military simulations are typically deployed within private networks, reliability and security concerns were outside the scope of this study. However, source code control, inherent to open source solutions, plays a central role in such considerations. Furthermore, many of these concerns stem from the absence of an organization responsible for managing the software lifecycle, which opens space for service-oriented business models similar to those adopted by companies like Red Hat.

In future work, a qualitative evaluation of the application of PORTICO in the integration of simulators for military exercises or various domains, such as industrial production lines, agriculture, urban mobility, energy, or smart cities, will be undertaken. Exploring automated code generation for the HLA interface is a promising research avenue that addresses challenges observed in this study and reported by other authors. This could improve efficiency and streamline the integration of the simulator system.

Building on these findings, this work establishes the groundwork for an MSaaS MVP, leveraging HLA standards to deliver scalable and efficient distributed simulation solutions for cross-domain applications. To achieve this, an infrastructure is being developed using the Kubernetes orchestrator, following a microservice architecture and employing JSON for data exchange. Integrating automation and HLA standards aims to simplify deployment and management, making simulation systems more accessible and adaptable.

6. Acknowledgements

This work was also partially financed by grants $\#2019/26702-8$, $\#2023/00782-0$, $\#2023/00811-0$ and $\#2024/01115-0$, from the Sao Paulo Research Foundation (FAPESP).

References

Amponsah, K. *et al*. (2019). Distributed building energy simulation with the HLA. In *Proceedings of the 2019 Summer Simulation Conference*. Society for Computer Simulation International.

Erciyes, K. (2019). *Distributed Real-Time Systems: Theory and Practice*. Springer Publishing Company, Incorporated, 1st edition.

Falcone, A. and Garro, A. (2018). Reactive HLA-based distributed simulation systems with RxHLA. In *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8. IEEE.

Falcone, A. *et al*. (2017a). An introduction to developing federations with the high level architecture (HLA). In *2017 Winter Simulation Conference (WSC)*, pages 617–631.

Falcone, A. *et al*. (2017b). Simplifying the development of HLA-based distributed simulations with the HLA Development Kit Software Framework (DKF). In *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–2. IEEE.

Grogan, P. T. and de Weck, O. L. (2018). Infrastructure system simulation interoperability using the high-level architecture. *IEEE Systems Journal*, 12(1):103–114.

Gütlein, M. and Djanatliev, A. (2020). Coupled traffic simulation by detached translation federates: An HLA-based approach. In *2019 Winter Simulation Conference (WSC)*, pages 1378–1389. IEEE.

Gütlein, M. *et al*. (2020). Performance evaluation of HLA RTI implementations. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8. IEEE.

IEEE (2010a). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Federate Interface Specification. *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000) - Redline*, pages 1–378.

IEEE (2010b). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*, pages 1–38.

IEEE (2010c). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification. *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000)*, pages 1–110.

Kuhl, F., Weatherly, R., and Dahmann, J. (1999). *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall PTR.

Malinga, L. and Le Roux, W. H. (2009). HLA RTI Performance Evaluation. In *2009 SISO European Simulation Interoperability Workshop*, pages 1–6. Simulation Interoperability Standards Organization.

Moller, B. *et al*. (2012). The HLA tutorial: A practical guide for developing distributed simulations.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann Publishers Inc.

Oliveira Rosa, J. R. (2023). HLA performance - github.

Pokorny, T. (2023). PORTICO - Runtime Infrastructure.

Pokorny, T. and Fraser, M. (2023). The PORTICO Project.

Russo, D. (2016). *Benefits of Open Source Software in Defense Environments*, pages 123–131. Advances in Intelligent Systems and Computing. Springer International Publishing.

Tanenbaum, A. S. and Steen, M. V. (2007). *Sistemas Distribuídos: Princípios e Paradigmas*. Pearson.

Technologies, P. (2023). Free HLA tutorial and software.

Tu, Z., Zacharewicz, G., and Chen, D. (2012). Developing a Web-Enabled HLA Federate Based on PORTICO RTI. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 2289–2301. IEEE.